CySecAgri

Design Document

Team 17
David Wolfe - IoT Sensors & Base station
Thomas Ruminski - IoT Sensors & Base station
Joe Hunter - Frontend Development
Syed Al-hussain - Frontend Development
Rian Lamarque - AWS Infrastructure
Eli Hanson - AWS Infrastructure

Contact Information
Email: sdmay23-17@iastate.edu
Website: https://sdmay23-17.sd.ece.iastate.edu

Client/Advisor
Dr. Manimaran Govindarasu

Revised: FIXME

Executive Summary

Project Objective

CySecAgri's primary goal is to develop an easily accessible closed loop platform that allows large and small farms to manage and track IoT sensor readings from all across their crops.

Development Standards & Practices Used

Flutter Application:

- ISO/IEC 5055:2021
- ISO/CD 18893

Cloud Computing:

- ISO/IEC 19944-1 (2020)
- AWS Well-Architected framework

IoT Sensors & Base station:

- IEEE 802.15.4
- IEEE 802.11ac
- LoRa Alliance Specifications

Summary of Requirements

Functional Requirements:

- IoT soil moisture sensor
    - Data collection
    - Medium-long range wireless transmitter (LoRa)
- IoT base station
    - Data logging
    - Medium-long range receiver to sensors (LoRa)
    - Wifi data transfer to AWS cloud
    - Secure and consistent data logging and distribution
- AWS cloud
    - Data logging for verified data integrity
    - Data graphing to display to farmers
    - Data transfer using MQTT for IoT and SQS for application
    - Use of Amazon S3, Lambda, IAM, and Elasticsearch to support data flow and security
- Flutter application
    - Secure login and sign up authentication for each user
    - Push notification alerts
    - Data visualization and representation of sensor data

    - Data logging and graphing of sensor data
    - Send and receive data from to the AWS cloud via Rest API

Non-Functional Requirements:

- IoT soil moisture sensor
    - Secure and consistent data collection and distribution
    - The IoT sensor will be able to withstand different climate conditions like rain, snow, and wind
- IoT base station
    - Secure and consistent data logging and distribution
    - The IoT base station will be able to withstand different climate conditions like rain, snow, and wind
- AWS cloud
    - Data can be stored and accessible from any location and time
    - Stored data will be secure
    - Deploy infrastructure via Terraform
- Flutter application
    - Will be available for both Android and iOS based devices
    - Will display sensor data to the user via two pages, a scroll view and a graph view
    - Users will only be able to see their sensor's data

Physical Requirements:

- Minimal Footprint (Constraint)
    - Can not take up space needed for crops
    - Can not impede tractor movement.
- Weather resistant (Constraint)
    - Water-proof
    - Hot and cold temperature tolerance
- High Visibility
    - Distinct colors to make it easy to locate in a field
    - Reflector strips at night and inclement weather

Resource Requirements:

- Batteries should last from planting to harvest
    - Minimal maintenance needed

Environmental Requirements:

- Batteries don't leak into soil
- Materials don't change soil nutrient levels

UI Requirements:

- Flutter application
    - Data is only accessible to platform user
    - System performs identically regardless of location
    - Application operates on multiple operating systems
    - Application can handle multiple users at the same time
    - Users can not view other users data

Security Requirements:

- IoT Sensors & Basestation

    - Data will be encrypted when stored on disk
    - Data will be encrypted end-to-end in transit
    - Sensors will be authenticated to the basestation using OTAA
- AWS Cloud
    - User roles will be properly separated
    - Minimum permissions will be allowed for all database accesses
- Flutter Application
    - Proper data sanitization will be enforced
    - Users will only be able to access their accounts data

Applicable Courses from Iowa State University Curriculum

- CprE 309 for creating the frontend and backend connections
- CprE 288 for working with UART and serial interfaces of IoT chips
- CprE 230/231 for understanding security principals
- CprE 489 for networking and data communication principles

New Skills/Knowledge acquired that was not taught in courses

- LoRaWAN connection protocols
- Flutter application design
- AWS API setup
- All AWS service knowledge

Table of Contents

Project Objective **1**

Development Standards & Practices Used **1**

Summary of Requirements **1**

Applicable Courses from Iowa State University Curriculum **3**

New Skills/Knowledge acquired that was not taught in courses **3**

Figures/Tables/Symbols/Definitions **7**

1 Team **8**

1.1 Team Members 8

1.2 Required Skill Sets for Your Project 8

1.3 Skill Sets covered by the Team 8

1.4 Project Management Style Adopted by the team 8

1.5 Initial Project Management Roles 8

2 Introduction **9**

2.1 Problem Statement 9

2.2 Intended Users and Uses 9

2.3 Requirements & Constraints 10

2.4 Engineering Standards 12

3 Project Plan **13**

3.1 Project Management/Tracking Procedures 13

3.2 Task Decomposition 13

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria 14

3.4 Project Timeline/Schedule 15

3.5 Risks And Risk Management/Mitigation 15

3.6 Personnel Effort Requirements 16

Table 1 16

3.7 Other Resource Requirements 17

4 Design **18**

4.1 Design Context 18
4.1.1 Broader Context 18
Table 2 18
4.1.2 Prior Work/Solutions 19
4.1.3 Technical Complexity 19
4.2 Design Exploration 20
4.2.1 Design Decisions 20
4.2.2 Ideation 20
4.2.3 Decision-Making and Trade-Off 21
Table 3 21
4.3 Proposed Design 21
4.3.1 Overview 21
4.3.2 Detailed Design and Visual(s) 22
IoT Sensors & Base station 22
Devices 22
Cloud Storage & Processing 23
Front End Application 23
4.3.3 Functionality 24
Initial Setup 24
Daily Use 24
4.3.4 Areas of Concern and Development 24
IoT Devices **25**
Cloud **25**
Front End **25**
4.4 Technology Considerations 25
LoRA RF and LoRaWAN 25
Amazon Web Services (AWS) 25
Flutter 26

4.5 Design Analysis 26

IoT Sensors & Base station **26**

Cloud **26**

Front End **26**

Overview **27**

5 Testing **27**

5.1 Unit Testing 27

5.2 Interface Testing 28

5.3 Integration Testing 28

5.4 System Testing 29

5.5 Regression Testing 29

5.6 Acceptance Testing 29

5.7 Security Testing 29

5.8 Results 30

6 Implementation **31**

7 Professional Responsibility **31**

Table 4 31

7.1 Areas of Responsibility 32

7.2 Project Specific Professional Responsibility Areas 33

7.3 Most Applicable Professional Responsibility Area 34

The most applicable professional responsibility area for this project would be sustainability. Our products will be deployed in fields that require constant maintenance and specific balances of nutrient levels. If our product does not respect this environment and take proactive steps to prevent damage, it will fail as none of our users will want to trade our product for the health of their crops. 34

8 Closing Material **34**

8.1 Discussion 34

8.2 Conclusion 34

Figures/Tables/Symbols/Definitions

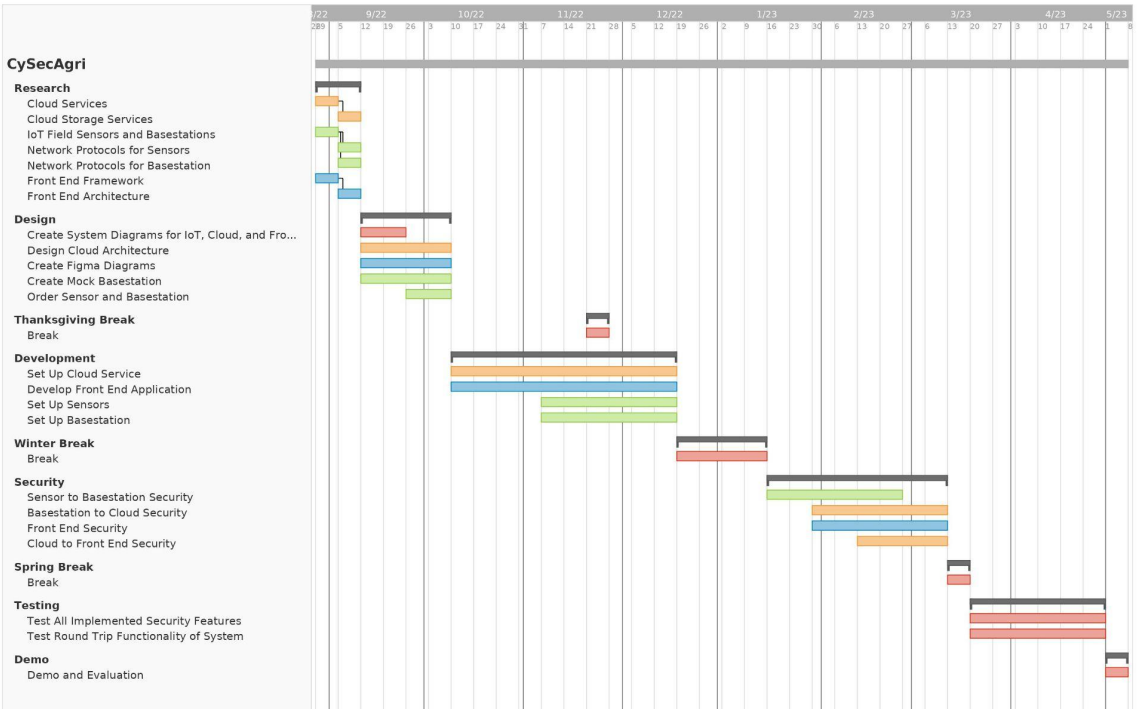


Figure 1. *Gantt Chart (Appendix A)*

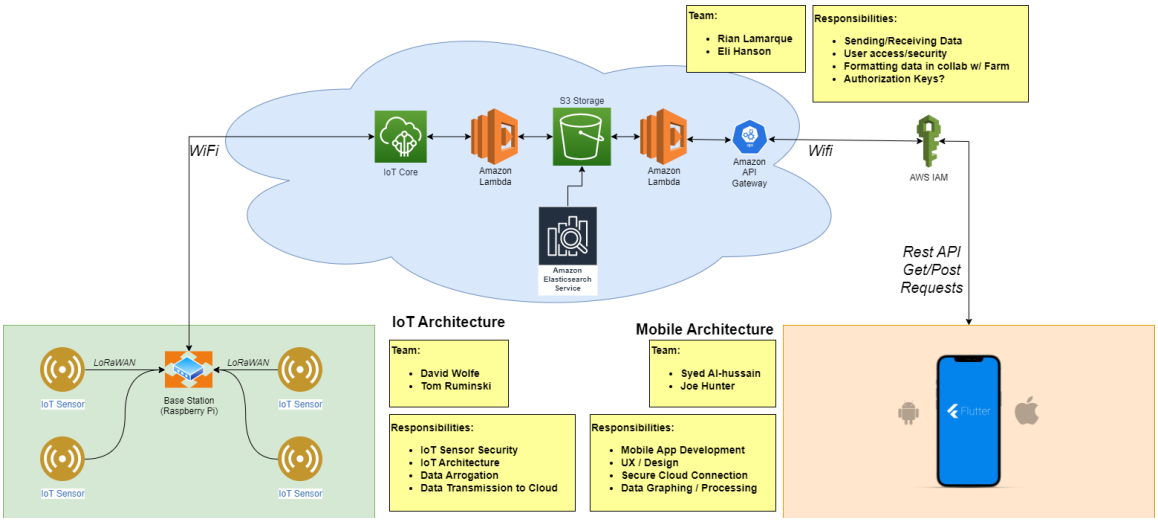


Figure 2. *System Diagram (Appendix B)*

1 Team

1.1 Team Members

David Wolfe, Tom Ruminski, Joe Hunter, Syed Al-hussain, Rian Lamarque, Eli Hanson

1.2 Required Skill Sets for Your Project

AWS service experience

Flutter framework experience

Lambda device management

GPIO programming

Python programming experience

1.3 Skill Sets covered by the Team

AWS service experience is covered by Rian Lamarque

Lambda device management is covered by Eli Hanson and Rian Lamarque

Python programming is covered by David Wolfe

GPIO programming is covered by Tom Ruminski

Flutter framework experience is covered by Joe Hunter and David Wolfe

1.4 Project Management Style Adopted by the team

Our team agreed to adopt an agile development project.

1.5 Initial Project Management Roles

David Wolfe - IoT Sensors & Base station, Meeting Note Taker

Tom Ruminski - IoT Sensors & Base station

Joe Hunter - Frontend development

Syed Al-hussain - Frontend development

Rian Lamarque - AWS Infrastructure

Eli Hanson - AWS Infrastructure

2 Introduction

2.1 Problem Statement

Our project is focused on solving the problem of professional farmers and hobby gardeners needing to manually test, record, and chart IoT sensor data. The solution we will work towards will include the farmers deploying IoT sensors and a base station to aggregate data. From that base station, data will be uploaded to the cloud for storage. We will then develop a front-end application for accessing this data. Our solution will include a number of different charts and graphs to help the users track crop progress and health over a selected period of time. We will also allow specific users to run different intrusion detection algorithms on the data that has been collected. These algorithms will produce alerts for the user if there are any issues with the data collection, if there are any data values that are out of the ordinary, or if there has been an indicator of compromise.

2.2 Intended Users and Uses

Intended Users:

1. Large Scale Farmers
2. Gardeners (Smaller family farms)
3. System Administrators
4. People who need to monitor plants from a distance (Corporate or Co-Op)
5. Owners of only house plants (Students)
6. People learning to take care of plants

Beneficial User Groups:

All the users listed above will benefit from our project since it will allow them to easily see the conditions of their plants and soil. Some of the benefits will include increased free time and less monetary overhead for monitoring their crops.

Intended User breakdown by persona and needs:

Large Scale Farmers Persona: Majority are white, older males. Their hobbies and interests include farming. They are motivated to make a living and feed the world. Their personalities are usually more business-oriented and value quantity and value of their produce.
Need For this Product: This product gives large-scale farmers the ability to collect more data about their field, get immediate alerts if problems occur, and track data trends over time.

Gardeners (Smaller family farms) Persona: Majority are male and own their family farm. Their hobbies include gardening and farming. Their motivations to do this are for retirement and their own personal interests. They are also very personally interested in the farm's success with a community focus.
Need For this Product: This product will allow gardeners to monitor water levels while away from the field, get notification reminders, and reduce time spent collecting data.

System Administrators Persona: SA's demographic would be technically oriented people making a living off this product. Hobbies for these people would be technology and agriculture. Their motivations would be to help farmers and earn a paycheck. Their personality would be very technical and problem-oriented. They would value customer satisfaction.
Need For this Product: This product will give System Administrators the ability to set permission levels for different users.

People who need to monitor plants from a distance (Corporate or Co-Op) Persona: These people would be businessmen or women who may not be on the farm too often. Their hobbies would include analyzing data, discovering new products and ideas, and being around people. Their work motivation would be business oriented and growing their farms. Their personality would be outgoing and personable. They would value company satisfaction and profit.
Need For this Product: This product will give Corporate/Co-Op users the ability to analyze their farms and fields over time and get consistent, immediate data from the fields.

Owners of only house plants Persona: This demographic of people would be homeowners or students who want to improve their living conditions with plants. Their hobbies are very open-ended, but all would at least have some interest in plants. Their work motivation wouldn't be very relevant since they aren't making money from our product. They could have a wide range of personalities since the demographic is so broad. They would value their time and ease of information.
Need For this Product: This product will give house plant owners the ability to monitor water levels while being away from the house and get notification reminders.

People learning to take care of plants Persona: This demographic leans toward students. Their hobbies would include learning about agriculture and horticulture. Their work motivation would be related to school and their own interests. Their personalities would be very broad, but all would be eager to learn about plants and soil conditions. They would value and have aspirations of bettering the field of agriculture and horticulture.
Need For this Product: This product will give people learning about plants the ability to monitor water levels, get notification reminders, and easily see what needs the plants have as well as how to fulfill those needs.

2.3 Requirements & Constraints

Functional Requirements:

- IoT soil moisture sensor
    - Data collection
    - Medium-long range wireless transmitter (LoRa)
- IoT base station
    - Data logging
    - Medium-long range receiver to sensors (LoRA)
    - Wifi data transfer to AWS cloud
    - Secure and consistent data logging and distribution

- AWS cloud
    - Data logging for verified data integrity
    - Data graphing to display to farmers
    - Data transfer using MQTT for IoT and SQS for application
    - Use of Amazon RDS, IAM, and Elasticsearch to support data flow and security
- Flutter application
    - Secure login and sign up authentication for each user
    - Push notification alerts
    - Data visualization and representation of sensor data
    - Data logging and graphing of sensor data
    - Send and receive data from to the AWS cloud via Rest API

Non-Functional Requirements:

- IoT soil moisture sensor
    - Secure and consistent data collection and distribution **(constraint)**
    - The IoT sensor will be able to withstand different climate conditions like rain, snow, and wind
- IoT base station
    - Secure and consistent data logging and distribution
    - The IoT base station will be able to withstand different climate conditions like rain, snow, and wind
- AWS cloud
    - Data can be stored and accessible from any location and time
    - Stored data will be secure **(constraint)**
    - Deploy infrastructure via Terraform and potentially Drone
- Flutter application
    - Will be available for both Android and iOS based devices
    - Will display sensor data to the user via two pages, a scroll view and a graph view
    - Users will only be able to see their sensor's data

Physical Requirements:

- Minimal Footprint
    - Can not take up space needed for crops **(constraint)**
    - Can not impede tractor movement **(constraint)**
- Weather resistant
    - Water-proof **(constraint)**
    - Hot and cold temperature tolerance **(constraint)**
- High Visibility
    - Distinct colors to make it easy to locate in a field
    - Reflector strips at night and inclement weather

Resource Requirements:

- Batteries should last from planting to harvest
    - Minimal maintenance needed

Environmental Requirements:

- Batteries don't leak into soil
- Materials don't change soil nutrient levels

UI Requirements:

- Flutter application
    - Data is only accessible to platform user
    - System performs identically regardless of location
    - Application operates on multiple operating systems
    - Application can handle multiple users at the same time
    - Users can not view other users data

Security Requirements:

- IoT Sensors & Basestation
    - Data will be encrypted when stored on disk
    - Data will be encrypted end-to-end in transit
    - Sensors will be authenticated to the basestation using OTAA
- AWS Cloud
    - User roles will be properly separated
    - Minimum permissions will be allowed for all database accesses
- Flutter Application
    - Proper data sanitization will be enforced
    - Users will only be able to access their accounts data

2.4 Engineering Standards

Flutter Application:

- ISO/IEC 5055:2021: Software quality standards for source code to be measured upon. The standard considers the reliability, security, performance efficiency, and maintainability of source code.
- ISO/CD 18893: Applies to mobile elevating work platforms to protect users from personal injury, property damage, and accidents. It also establishes criteria for inspection, maintenance, and operation.

Cloud Computing:

- ISO/IEC 19944-1 (2020): Provides guidance on ensuring data flow and use as well as making cloud service agreements. Helps guide transparent use of data across the cloud services. Doing so will maintain trust between use parties and protects personally identifiable information (PII)
- AWS Well-Architected framework: Defines six pillars to help create the best design possible in the cloud. Assists with efficiency, cost-optimization, and sustainability.

IoT Sensors & Base station:

- IEEE 802.15.4 is a standard that defines the operation of a low-rate wireless personal area network (LR-WPAN). It specifies the physical layer and media access control for LR-WPANs. Zigbee is regulated under 802.15.4 and is the communication method between nodes and the base station.

- IEEE 802.11ac is the most recent update to ac wi-fi standards. Our devices may use wi-fi, so following the IEEE standards will be important for compatibility purposes.
- LoRa Alliance is the governing body controlling specifications for the three classes of LoRa devices (A, B, C). It controls regional frequency bands, data rates, and security standards for LoRa devices.

3 Project Plan

3.1 Project Management/Tracking Procedures

Our team plans on using an Agile project management style with the addition of a Gantt chart to help keep track of long-term goals. This is the most effective way to manage our team due to the fact that each aspect of our project is closely connected to one another. In order for the AWS team to complete their weekly goals, the IoT team must also be making strides in goals related to sending data to AWS. The same goes for the front end requiring AWS data to be used for visualization and display. Having quick weekly meetings and Agile-like sprints will allow us to ensure that teamwork is happening and that nobody is being seriously bottlenecked by anybody else.

The use of a Gantt chart on top of our standard Agile development will be useful to ensure we aren't spending more time than we have on specific aspects of the project. Each of the components of our system will require constant tweaking and development. However, we plan to track several major milestones with our Gantt chart. These milestones include a full connection between our IoT base station and AWS, a full connection between AWS and our frontend application, and different data visualization goals.

The software we plan on using to manage these goals is a combination of GitLab for code management and version control as well as a shared KanBan board hosted on Mattermost to help keep our teams working efficiently and following our Agile development goal.

3.2 Task Decomposition

Our team has broken all major projects down into more manageable tasks that can be delegated to individual team members. These decomposed tasks are shown below. They are grouped by overarching sprint goal. These tasks are also displayed in Figure 1: Gantt Chart.

- IoT Base Station connection to AWS
    - Being able to send and receive data securely
    - Storing data securely and making it easily accessible
    - Ensure access to user-specific data
- Flutter App connection to AWS
    - Being able to send and receive data securely
    - Allow transmission of data to legitimate users
- Frontend data visualization
    - Correctly displaying our data in a way that Farmers can understand and use
- Connect the IoT Sensors to the Base Station

    - Setting up LoRaWAN protocol to connect the sensors and base station
- Secure the AWS Cloud
    - Limited permissions
- Securing the IoT sensors and Base Station
    - End to end encryption
    - Exhaustive penetration testing
- Secure the Flutter App
    - End to end encryption
    - Exhaustive penetration testing

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Major milestones for our project are grouped by large features in the end application. We've included specific and measurable metrics for success to ensure each of these milestones are accomplished to a satisfactory level.

- **Round trip data connection from IoT sensor to frontend**
    - Metric: Live updates when new sensor readings come in
    - Metric: Delay of no greater than 5 minutes
- **Secure login system for frontend**
    - Metric: Offensive security testing is unable to find flaws in login page
- **AWS data logging**
    - Metric: Data stored in the cloud
    - Metric: Data separated by user/group
- **Data analysis implemented in the cloud**
    - Metric: Data is being fed into AWS machine learning algorithm
    - Metric: Accurate assessments of the input data is returned from the algorithm
- **Front end data graphing**
    - Metric: Users are able to see useful visualizations of their data

3.4 Project Timeline/Schedule

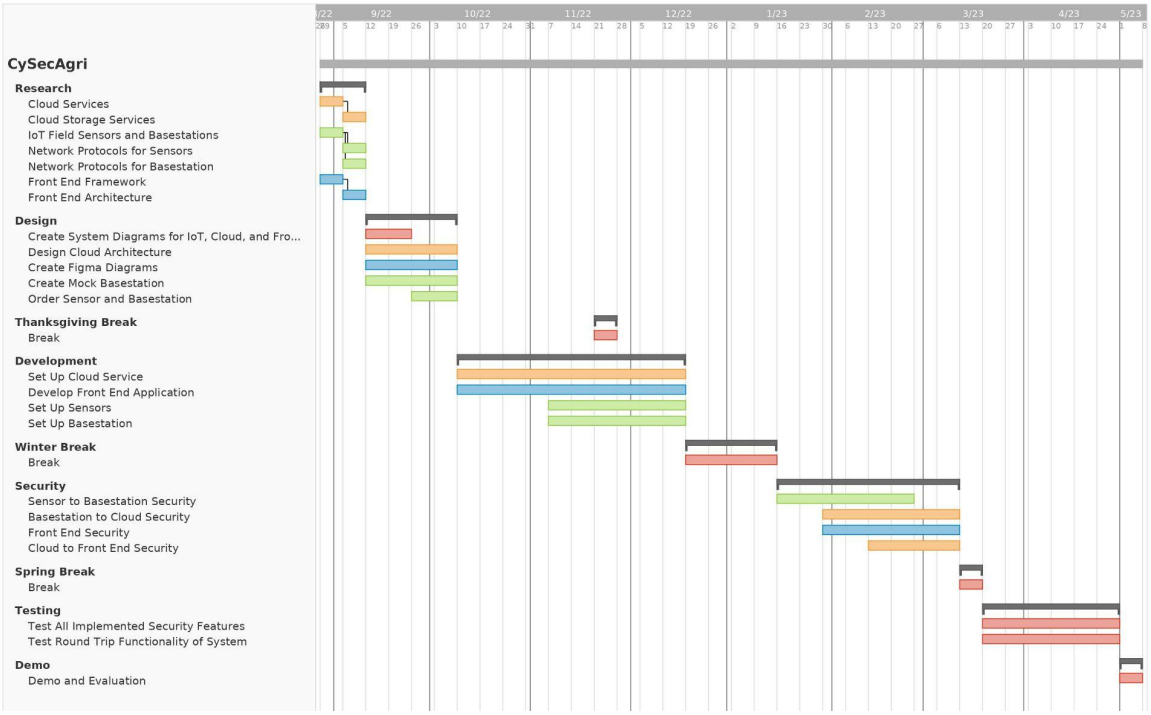


Figure 1. *Gantt Chart (Appendix A)*

3.5 Risks And Risk Management/Mitigation

Consider for each task what risks exist (certain performance targets may not be met; certain tools may not work as expected) and assign an educated guess of probability for that risk. For any risk factor with a probability exceeding 0.5, develop a risk mitigation plan. Can you eliminate that task and add another task or set of tasks that might cost more? Can you buy something off-the-shelf from the market to achieve that functionality? Can you try an alternative tool, technology, algorithm, or board?

Agile projects can associate risks and risk mitigation with each sprint. The following table displays the risks associated with our project and the assumed severity based on the table in Apendix E

Risk	Severity
Cannot connect our base station to AWS	9 - Moderate
Cannot connect our flutter app to AWS	3 - Minor
Cannot use the data that is received	3 - Moderate
AWS shuts down	1 - Minor

The servers and base station are incompatible with each other	6 - Moderate
The sensors are proprietary so we cannot do any securing	15 - Major
The flutter app won't be able to be secured because it will not be set up properly	4 - Minor

Risk Mitigation Plan: If the sensor ends up being proprietary and is unable to be secured by us, we will have to drop this task. It would be too far into the project to find a different sensor and connect it to our system. So, if this were to occur, we would add a task to replace this task where we do our sensor data validation and security in the cloud instead.

3.6 Personnel Effort Requirements

Table 1

	David Wolfe	Tom Ruminski	Joseph Hunter	Rian Lamarque	Eli Hanson	Syed Al-Hussain	Total Hours:
Base Station to AWS Connection	**8**	**2**	**0**	**18**	**8**	**0**	**32**
Flutter App to AWS Connection	**0**	**0**	**10**	**4**	**8**	**4**	**26**
Frontend Data Visualization	**0**	**0**	**16**	**4**	**4**	**10**	**34**
Secure AWS Cloud	**0**	**0**	**0**	**12**	**16**	**0**	**28**
Connect Sensors to Base Station	**4**	**8**	**0**	**0**	**0**	**0**	**12**

Secure IoT Sensors	**10**	**12**	**0**	**0**	**0**	**0**	**22**
Secure Base Station	**14**	**14**	**0**	**0**	**0**	**0**	**28**
Secure Flutter Application	**2**	**0**	**10**	**0**	**0**	**8**	**20**
Total Hours:	**38**	**36**	**36**	**38**	**36**	**22**	**206**

3.7 Other Resource Requirements

SenseCAP S2104 - LoRaWAN® Wireless Soil Moisture and Temperature Sensor:

- Cost: $130 (x2)
- 2 Sensors
- Battery Type: Standard D-size
- LoRaWAN Device Class A (least power consuming)
- No required modifications

Base Station:

- RaspberryPi Model 4B- x1
  - Cost: $180 (market price)
- LoRaWAN Concentrator Chip RAK2245 Pi HAT- x1
  - Cost: $120
- 915MHz (8dBi) Antenna
  - Cost: $65

4 Design

4.1 Design Context

4.1.1 Broader Context

Table 2

Area	Description	Examples
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., a solution is implemented in their communities)	Increasing/reducing exposure to pollutants and other harmful substances, increasing/reducing safety risks, and increasing/reducing job opportunities.
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	The development or operation of the solution would violate a profession's code of ethics, and implementation of the solution would require an undesired change in community practices.
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.	Increasing/decreasing energy usage from nonrenewable sources, increasing/decreasing usage/production of non-recyclable materials.
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	The product needs to remain affordable for target users, the product creates or diminishes opportunities for economic advancement, and high development cost creates risk for the organization.

- Public health, safety, and welfare
    - IoT systems reduce the need for manual data collection, which can be physically taxing. This will increase safety for those who are injured or otherwise incapable of performing this task.
    - Securing our devices means increased safety and privacy for those who operate them.
    - Increased agricultural data collection and analysis could lower the barrier to entry for people interested in pursuing a career in agriculture.
    - IoT smart agriculture could increase crop yields and improve food security globally.
- Global, cultural, and social
    - Through securing agricultural IoT devices, we can increase trust with farmers and the general public.
    - NIST security standards for IoT systems will need to be of utmost importance.
- Environmental

    - IoT sensors and gateways can be powered by renewable energy. As technology improves, our design can be adapted to renewable energy sources.
    - Without the need to produce batteries, less toxic pollutants will be released as a byproduct of production.
    - Our IoT devices are considered low-power and use as little energy as possible.
    - By taking advantage of distributed sensor networks, agriculturalists can better target specific field areas that may need additional resources instead of applying potentially harmful materials throughout the entire field.
- Economic
    - Product cost can be tailored to consumers depending on their needs (e.g. a small farm will require fewer sensors and cloud resources than a corporate farm).
    - Our IoT security implementations will increase trust in such systems and could expand their adoption into more farms.
    - With real-time access to sensor data in their fields, farmers could potentially increase their yields which might reduce the cost of food to consumers.

4.1.2 Prior Work/Solutions

A. Mishra, "Cloud-based multi-sensor remote data acquisition system for precision agriculture (CSR-DAQ)," Thesis, 2020.

- Mishra details a cloud infrastructure using MQTT as the key protocol for data transmission to and from the cloud. The primary focus of the paper was on cloud architecture. Our design will likely be similar at the cloud level but will be a full physical implementation of an IoT system with data collection on soil moisture and temperature.
    - Our work will also be a combination of the LoRa RF/LoRaWAN protocols and MQTT. LoRa is our physical layer transmission from our sensor to our gateway and was not a component of Mishra's design
- Mishra created a web application as well as an android application for data representation to the customer. The web application was developed using Python and the Flask framework, and the android application using Java and Android Studio. Both were connected to AWS through an Elastic Beanstalk Container and Amplify, respectively.
    - Instead of creating a web application, our group decided to create a mobile application. We will design a Flutter application, which will allow us to build an iOS and Android application simultaneously. Our application will also be connected to AWS via Amplify.

4.1.3 Technical Complexity

Our design satisfies the following two benchmarks for technical complexity:

1. Our design consists of multiple components/subsystems that each utilize distinct scientific, mathematical, or engineering principles

2. The problem scope contains multiple challenging requirements that match or exceed current solutions or industry standards.

These benchmarks are met in the following ways:

- At an infrastructure level, our system combines a variety of network protocols. We will be using LoRaWAN for communication between sensors and gateways, the TCP/IP suite for communication with the AWS IoT Core, and AWS IoT Core's set of rules for communication with AWS Cloud Services
- After implementation, extensive testing will be performed on security protocols between sensors, gateways, the cloud, and our user application
- We will be employing different kinds of hardware/computing elements. LoRa capable sensors, a RaspberryPi with a LoRaWAN transceiver, and AWS Cloud Services will be our core components.
- Our cloud infrastructure will be built using the AWS services mentioned above. Each service will require appropriate scoping of permissions, cost optimization, and full functionality testing. The infrastructure will be deployed to AWS via terraform and a GitLab CI/CD pipeline. Services used will include Lambda, DynamoDB, IoT Core, SQS, and IAM.
- Challenging requirements include:
    - LoRaWAN to MQTT conversion at our gateways
    - Low-power authentication for sensors and gateways
    - Data analysis and anomaly detection at the cloud level
    - UX design for farmers viewing their data through our application

4.2 Design Exploration

4.2.1 Design Decisions

The following are three crucial design decisions we have made:

1. We chose a star topology and LoRa RF physical medium for sensor-to-gateway communication. This is less complex than mesh-network systems but reduces weak points in our future security protocols. LoRa RF was chosen as our communication medium because it operates on low power and has a range of ~1km. This decision will enable us to better protect against "imposter" attacks and implement fewer nodes across a large area.
2. We chose Flutter to develop our user interface. We chose Flutter because it reduces development time yet increases access to our application. This is possible due to Flutter providing the ability to have an iOS and Android application be generated from a single code base. With Flutter we only have to develop a singular application, reducing development time, which can be used by both major mobile platforms, increasing access.
3. Cloud computing platforms were another area where our team had to reach a decision. We decided to use AWS to service our cloud computing needs. This decision is important to our project's success because cloud computing can be difficult to learn. If the platform we chose did not provide the necessary tools to complete our project, we would be at a loss of significant development time.

4.2.2 Ideation

One decision that needed to be made was: deciding which front-end framework will we use to develop our application. To make this decision we used the ideation process of brainstorming. From

our brainstorming, we came up with five different frameworks that we could use to build our application. Those frameworks were:

- React Native
- Android (Java)
- Flutter
- IOS (Swift)
- Creating an application for all platforms, ie. web and mobile

After we brainstormed these five different choices, we decided to research each choice to see which framework would work best for our project. This research step took about a week and once we had finished it we moved on to our next step which was creating a decision-making matrix.

4.2.3 Decision-Making and Trade-Off

From our matrix, Flutter was the clear decision for us. Our criteria included how much time would be spent developing in comparison to our other options, how much experience we have with each option, and how easy it would be for our users to access our application. Flutter was tied in the time category as most other options would require the same amount of development time. Flutter was also tied for first in the experience category as our team as a whole had the same amount of experience with Flutter as React. Where Flutter took the lead was in the access category. With Flutter we would be able to create an application for all major mobile platforms, allowing our users to easily access our application from their smartphones. This made Flutter our best option to use as a front-end framework.

Table 3

Selection Criteria	Criterion Weight	React		Android		Flutter		IOS		All	
		Score	Total	Score	Total	Score	Total	Score	Total	Score	Total
Time	.25	3	.75	3	.75	3	.75	3	.75	1	.25
Experience	.5	3	1.5	2	1	3	1.5	1	.5	1	.5
Access	.25	3	.75	2	.5	4	1	2	.5	5	1.25
Total	1	9	3	7	2.25	10	**3.25**	6	1.75	7	2

4.3 Proposed Design

4.3.1 Overview

Our project is focused on solving the problem of professional farmers and hobby gardeners needing to manually test, record, and chart IoT sensor data. The solution we will be working towards will include the farmers deploying IoT soil sensors and a RaspberryPi base station to aggregate data. From the base station, data will be uploaded to Amazon Web Services for storage and processing.

Users then access this data and information from our front-end application. This front-end app will give the user access to different charts and graphs to help the users track crop progress and health over a selected period of time. We will also allow specific users to run different intrusion detection algorithms on the data that has been collected. These algorithms will produce alerts for the user if there are any issues with the data collection, if any data values that are out of the ordinary, or if there has been an indicator of compromise from a cyber security perspective. Below is a more detailed breakdown of the individual components and technologies we plan on using for our first deployment.

4.3.2 Detailed Design and Visual(s)

Our senior design system consists of three key components: IoT Sensors & Base station, cloud storage & processing, and the front-end application.

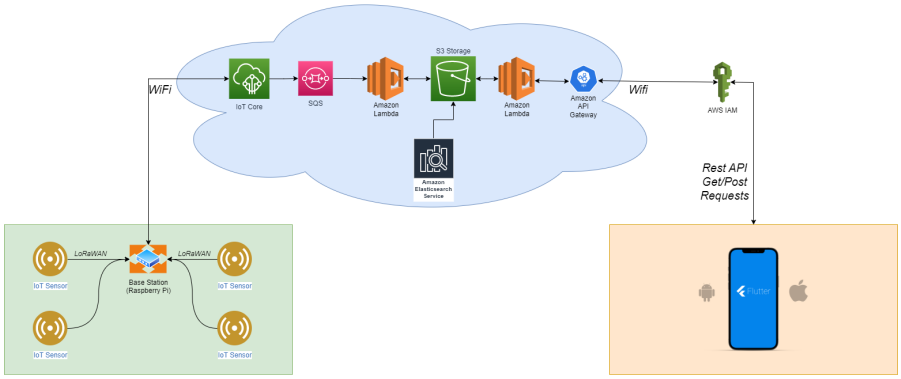


Figure 2. *System Diagram (Appendix B)*

IoT Sensors & Base station

Starting with the IoT Sensors & Base station, we have two sensor nodes connected to a base station in a star network topology. This base station will connect to a network server (AWS IoT Core) which will then interface with other AWS services utilized by our Cloud Development team. This system will be scalable by adding more base stations and nodes to the design.

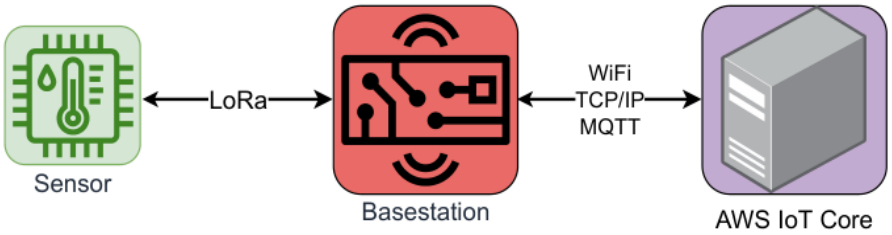


Figure 3. *IoT Sensor and Base Station Diagram.*

Devices

We have purchased two SenseCAP S2104 sensors which collect soil temperature and moisture data. These sensors are LoRaWAN enabled and will broadcast data to the base station. The base station is a RaspberryPi with a LoRaWAN receiver connected to it. The RaspberryPi is WiFi enabled and will

connect with AWS IoT Core via TCP/IP. AWS IoT Core is responsible for handling uplinks and downlinks from the sensors to the AWS Services the cloud team is using.

Included in Appendix C is a sample of the data format being sent from the sensors to our base station. This data is then validated using the CRC-16/Kermit algorithm to ensure no data was corrupted between the sensors and base station.

Cloud Storage & Processing

Currently, the cloud storage option uses s3 with specified path names for each farm. The data travels from the base station to the IoT Rule, which passes the message through a Simple Queue Service, reaching a Lambda function to break down the message. The Lambda function breaks down the message per hex values and converts them to decimal values that are then pushed into the database. In the future, this process may include the AWS ELK machine learning stack to ensure the data is validated.

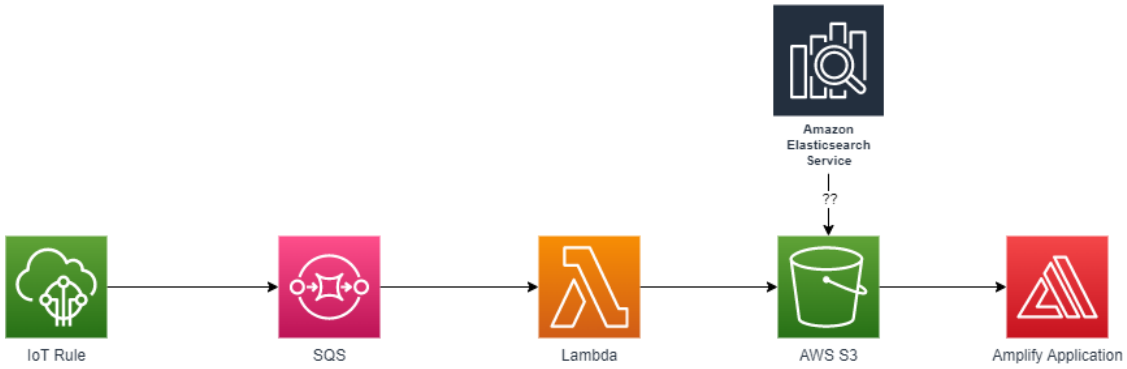
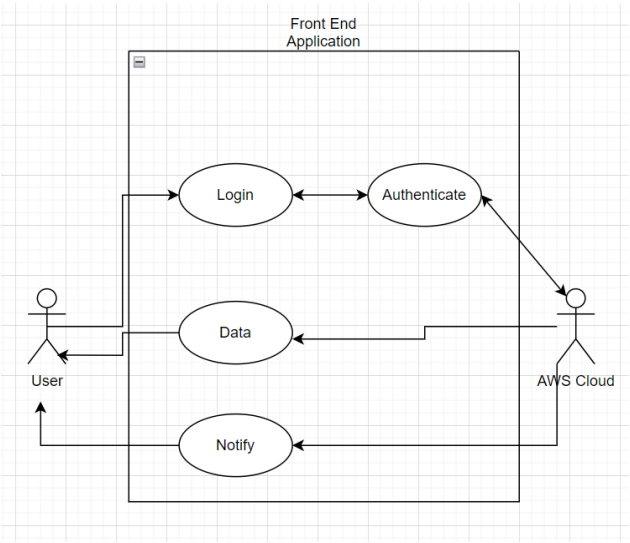


Figure 4. *Cloud Storage & Processing Diagram.*

Front End Application

For our application we decided to use a Flutter application to build our application layer. After researching multiple different ways to develop, including making a decision matrix, we came to the conclusion that Flutter would be our best choice for development. To connect our app to Amazon Web Services we used AWS Amplify, a backend service that connects our frontend Flutter app to AWS. To connect to our backend data we will use Rest API that will allow for easy and efficient communication. We will use multiple different Widget libraries to build our application, including AppBar, BottomNavigationBar, and Containers. These libraries will allow us to build a quality application that looks modern and performs efficiently. We will display information about our sensor on multiple different pages that we will design



for our users. These pages will be pre-designed using Figma.

Figure 5. User & AWS flow diagram

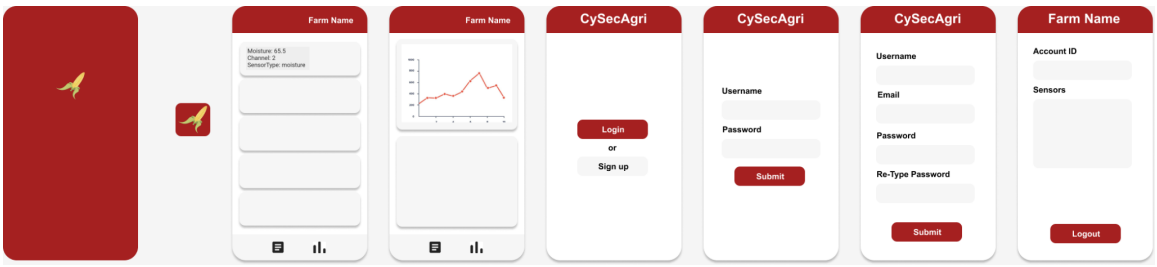


Figure 6. *Figma designs for app pages (Appendix D).*

4.3.3 Functionality

Initial Setup

Our end users would purchase our solution as a kit with a basestation and select sensors included. They would then place these sensors in different positions around their farm, garden, or home. Users will then download our front-end application to initialize their sensors for monitoring. Users will be able to name all of the sensors they have placed. There will be two different ways for users to view their sensor data. The first will be a scroll view that shows all the most recent sensor values as a scrollable list. The second way will be through a line graph where users will be able to see trends over time from their field.

Daily Use

After the initial setup users will simply have an application on their phone, tablet, or computer that allows them to monitor the sensor's reading at any time of day from anywhere. Users will open the mobile application and be greeted with a list of the different sensors and locations they are monitoring. From there they will be able to tap into any specific sensor and get live readings of the sensor values as well as a graphical representation of the sensor readings from the last week. This time scale for historical data will be customizable by the user.

4.3.4 Areas of Concern and Development

The current design satisfies physical requirements by keeping a small profile in sensor distribution. Sensors can be sparsely placed due to the use of a LoRa RF, star topology rather than a mesh network. The sensors we purchased are also weather and temperature resistant with a rating between -40 and 185 degrees Fahrenheit. This will be sufficient for a majority of climates, including Iowa.

Our primary concern for our user and client needs is security. Security is very important to both our clients and our user. We know we can meet the main functionality of our product by delivering

data from our sensors to user devices. Securing the data being transferred will be a little trickier. We must secure our data on all three levels of our product (Hardware, Cloud, User Interface), each of which will require unique security solutions.

IoT Devices

Currently we use CRC16/Kermit to ensure data integrity between our IoT sensors and the base station. We also employ asymmetric encryption between our base station and AWS to ensure data confidentiality when transmitting across the internet. Each sensor has a unique key and EUI that can be registered to the base station to help verify sensors. These sensor nodes are simple to install, and farmers would not need a technician to add more sensors to their farms.

Cloud

To mitigate cloud security concerns we plan to implement AWS Identity and Access Management policies so that users will not have unlimited privileges and are able to see other users' data. Also, we plan to use AWS Cognito to handle user authentication, this will help to make sure that only authorized users are able to access our cloud resources.

Front End

The front end will employ proper data sanitization to ensure users' security. On top of only allowing a strict set of characters to be entered for any user input field user password and accounts details will be stored securely in AWS. This includes hashing passwords and transmitting all data over secure, encrypted channels.

4.4 Technology Considerations

We have made key technology choices within each of our subteams. This includes the protocol stack between gateways and sensors, our cloud service, and front end application framework.

LoRA RF and LoRaWAN

At the perception level, Sensor-Base Station communication is performed through LoRaWAN and LoRa RF. The link layer and physical protocol have a key advantage over most other low power transfer options: range. Bluetooth Low Energy (BLE) and Zigbee are both popularly used in IoT applications for their low power usage. However, these protocol stacks are typically used in a mesh network configuration where a long signal range can be overkill. Limited to a maximum of 100 meters, BLE and Zigbee are not suited for the sparse dispersion of sensor nodes on a moderately large farm. Enter LoRa RF. LoRA RF has similar low-power usage at a significantly longer range of 1km. LoRaWAN is the accompanying link layer to LoRa RF, and our purchased sensor node and transceiver (for the base station) are already LoRaWAN enabled. In addition, LoRa devices cost about the same BLE or Zigbee making it financially viable for our project. In conclusion, LoRa enabled devices were chosen for their low-power usage, long range, and affordability.

Amazon Web Services (AWS)

There were three main options we thought about when choosing our cloud platform, AWS, Google Cloud, and Azure. We chose to go with AWS for a number of reasons. One of them was the fact that

one of the members of the group is very familiar with AWS. Also, our client already had an AWS account, so it would be simple to give us access to the platform. Our client also mentioned the idea of using machine learning to implement an intrusion detection system, and AWS has a service that supports that functionality called ELK. We briefly entertained the idea of using other cloud platforms, but it quickly became clear that AWS was the best fit for our project due to previous experience and the services it offers.

Flutter

When first considering what we wanted to create for farmers, we considered multiple different options. At the start, we debated whether we wanted a web application or a mobile application. After talking to farmers about their wants we decided that we would go with a mobile application. Once we decided on a mobile application, we had to figure out how we would create our mobile application. We looked at mobile frameworks that were easy to develop for both Android and iOS operating systems. And after making a decision matrix we ended up going with a Flutter mobile application. We chose Flutter because it is a fast reliable framework that can simultaneously develop an Android and IOS mobile application.

4.5 Design Analysis

IoT Sensors & Base station

Thus far we have ordered and received the two IoT soil temperature and moisture sensors, the RaspberryPi 4, and the LoRaWAN breakout board/antenna. Our IoT team spent a week testing this initial IoT setup and found a small issue with our chosen breakout board. While testing with the RAK3172 breakout board we noticed we were able to send LoRa messages but were unable to receive them. Upon further investigation, we discovered that the RAK3172 lacks a proper LoRaWAN concentrator chip. This is the chip responsible for allowing our breakout board to act as an aggregator for other LoRaWAN equipped devices like our sensors. This was a small road bump for us and something we overcame by placing an additional order for a RAK2245 Pi Hat. This Hat will sit on top of our RaspberryPi and enable all of the features required for our base station. We are currently waiting for this part to arrive, after which we will resume testing.

Cloud

In the cloud, we've implemented an AWS Lambda function to take in that simulated data from the base station and parse it out into the proper JSON format according to the data sheets for our sensors. We've also set up a simple DynamoDB for storing our sensor reading, this may not be our final design. Additionally, we have another Lambda function that will pull data from our database to send to the user. Utilizing AWS's API service we have been able to connect our front-end mobile application to our AWS data. This API returns a JSON array of the relevant sensor measurements and IDs for the users currently logged into the mobile application.

Front End

In the beginning, front-end development has been focused on establishing a connection between the app and AWS. Getting that established connection took longer than expected. We had originally

planned to use AWS's Amplify service, but after spending the timespan of a week and a half with no success, we decided to ditch Amplify and create our own RESTful API system to connect to AWS. Once that was complete, our attention was turned to displaying the sensor data in a user-friendly way. At the moment, we have two views for the user that are currently in progress. The first view is called our scroll view which allows users to see every sensor measurement in a list of scrollable containers. Our second view is a graph view that is still being developed but will display a line graph so that users can see trends from their soil sensor over time.

Overview

One issue we've run into that has not yet been fully fleshed out is how we plan on storing data in AWS. As the type of data we will be storing and what format it will be in is still up in the air, it's hard to commit to one specific mode of storage. Currently, we are building out a DynamoDB table. However, we've also considered using AWS's RDS service.

We are also being very cautious about the types of IoT radios and sensors we are selecting for the project. There are a lot of subtle differences between the parts and we need to make sure they all work together. These issues will become harder to resolve as the semester goes on as part orders can take several weeks to fill.

5 Testing

5.1 Unit Testing

Our CRC-16/KERMIT checking class will be tested for accurate packet validation. This will be tested by using a distribution of valid and invalid packets to give the class. CRC validation needs to operate at 100% accuracy, so evaluating accuracy is actually rather simple.

Inside our AWS infrastructure, ensuring that the data is passed between services is crucial. By running basic test payloads in the AWS console, we can see if the data is successfully passing through. Our packet analyzer lambda needs to receive information from the IoT Core through the SQS so we can run our mock base station script to send information to the IoT Core. From there, using CloudWatch to view the logs, we can see whether the data reaches the analyzer lambda successfully or what errors may have occurred. As well as testing in the cloud, these scripts can be tested locally to see how they react because we know what the packet format will look like.

Additionally, our front-end application will incorporate unit testing to ensure the accuracy of what we are designing. The Flutter framework has a nice directory in the main application directory to create and run unit tests called /test. All that is needed to do to be able to run unit tests is to add the test or flutter_test dependency to the pubspec, Flutter's package manager file. To add even more functionality to our unit tests, we will also use the Mockito dependency. With unit testing enabled for our front-end application, we will be able to make sure data can trigger certain functionalities and features in our app. For example, we could mock up a JSON value and make sure the mocked JSON is displayed correctly on the screen and returns the correct response code. For every functionality of our application, we will include a unit test to verify the correctness of the functionality.

5.2 Interface Testing

Our two biggest interfaces for this application would be our IoT sensor network and our AWS infrastructure. Due to the interconnectedness of these two interfaces, it will be essential to not only test them in their own unit test but also test them together as one large interface. These tests will include reverse engineering our base station code and attempting to send malicious or malformed packets to it from our sensors. If we are able to get one of these modified packets through the base station it will be put into our AWS database and could potentially cause more damage to the stability of our application.

Just as we test the data flow from our IoT sensors to AWS, it will be important to understand the ways in which data from AWS can affect our sensor network. We will be attempting to implement a naming system where users of our application can name and digitally place sensors over a map image of their land to better keep track of the positions their readings are coming from. This feature will require data to be sent from our user configuration database in AWS to the sensor and interact with them. This will require a large amount of manual testing to ensure AWS is not able to put the sensors in some state that is unrecoverable or requires physical interaction with the device (hard resetting it).

5.3 Integration Testing

One significant integration path on the hardware end of our IoT system is Sensor-to-Base Station communication. Due to constructing our own base station, a lot of testing will need to be completed on both communication paths of the base station to ensure our implementation is functioning properly. Due to the nature of the LoRaWAN protocol, we can expect about a 10% packet loss rate as a reasonable amount for our system and its applications. Communication testing outside with obstacles (or the actual crops they're meant to simulate) should be completed for reliability. If our packet loss rate is consistent with the 10% margin, it will be considered suitable for our agricultural application. Since the packet structure created by our sensors have a CRC code attached to it, accurate transmission can also be examined.

The next integration path is between the base station and the AWS IoT core. This will operate over the TCP/IP stack and use an MQTT application protocol. Reliability testing will need to be completed at this stage to evaluate the functionality of our MQTT implementation. Again, due to constructing our own base station, testing will be more focused on the base station and its configuration with the AWS IoT Core. Due to our IoT system's infrequent data collection, speed and latency are not as important to consider during testing. MQTT (particularly Quality of Service 2) is considered very reliable, and our pathway will need to be held to that standard. To test this, we can manually disrupt the network connection of the base station and evaluate how it recovers. If, after the disruption, the base station still accurately sends its data and it is properly received by the IoT core, then our system can be considered reliable for data transmission to the cloud.

The final integration path will be between the AWS API gateway and our Flutter front-end application. This connection will be made possible through a Rest API gateway provided by AWS and received by our front-end application. To test the functionality and correctness of this connection, we create integration tests in our front-end application. These tests will be similar to how our unit tests for our front-end application will be implemented. We will add the Flutter integration_test dependency to our pubspec, and after that, it will be as simple as running the tests

we create to see if they pass. This will allow us to test the functionality of our application to see functionality like if buttons are being triggered correctly and pages are being displayed properly.

5.4 System Testing

Our full loop system consists of three smaller subsystems in the IoT sensors, AWS infrastructure, and frontend application. For system testing, we will need to ensure that we can have data transferred through our entire system. Once we can do that, we will be able to see how the system interacts as a whole. This form of testing will be manual in nature because of the size of our system. Each form of testing prior to this will be integral to getting a foundation for our system testing. Unit testing will allow us to look at a specific aspect of our system, interface testing will help us see how two parts connect to each other, and integration testing will show us how those paths of dataflow in our system move.

5.5 Regression Testing

Since many of our development changes will be centered on the base station, consistent regression testing to determine that new security additions have not broken our sensor-base station-cloud communication path for uplinks and downlinks. A few uplink and downlink transmissions should be sufficient for smaller modifications and can be expanded for larger design improvements for more thoroughness.

The next way our project will regression test is by creating a Continuous Integration and Continuous Deployment pipeline. This will allow us to test our new code on our old unit and integration tests. We will create these CI/CD pipes in Gitlab, and once they are set up, they will run every time we push our code to Gitlab. If one of our old tests fails, the CI/CD pipeline will also fail, indicating to us that there has been a regression. This will help us with the AWS side of our project as well as in our front-end application.

5.6 Acceptance Testing

We plan on keeping track of whether the test was successful or not and, if not, what specifically the error was. In order to monitor which areas of our project we have tested, we will categorize the attacks and mitigation techniques based on the relevant area of the project that they apply to, such as IoT devices, AWS infrastructure, or frontend applications.

To make sure that our client is aware of our testing progress, we have been providing live demonstrations during our weekly meetings. Going forward, we will continue this but also increase communication so that our client is always aware of our progress. Our client has stressed that for a successful project, we should ensure that the data communication is sent securely from the sensors to the user through Amazon Web Services. To make sure that our client's needs are satisfied, we focus not only on securing the inter-component communication but also making sure data is secure inside each component, such as information stored in databases or adding more sensors.

5.7 Security Testing

Significant security testing will need to be completed as a part of our project's objective. For Sensor-to-Base Station communication, three key attack vectors will need to be examined: packet sniffing, packet modification, and imposter sensors introduced to our system. Packet sniffing can be mitigated by the standard LoRaWAN protocol's use of AES-128 bit encryption. Verification testing

can be done to ensure all packets are properly encrypted with AES-128. Packet modification can be mitigated by the aforementioned AES-128 bit encryption and the CRC-16/KERMIT verification. If a packet is modified, the base station will be able to verify the validity of the data by calculating and comparing the CRC code for each packet. This can be done by passing a mix of accurate and modified packets to the base station to track its ability to evaluate data validity. Imposter sensors are trickier to test. LoRaWAN specified Over Air Authentication (OTAA) as the join procedure. OTAA is already well established as a protocol so testing will revolve around the implementation of the protocol rather than its validity. To test this, we can introduce an unregistered sensor into the range of the base station and test whether it will be able to connect/authenticate with the base station without our approval or knowledge.

There will also be significant security testing for the connections between our front-end application and our AWS infrastructure. This will mostly include manual penetration testing of our mobile application and our API. We will use industry-standard tools BurpSuite and SQLMap for testing our login system. This round of testing will ensure proper mitigation of SQL injection and authentication bypass. After this, we will also use Postman and BurpSuite to see exactly what data our application is processing. We will then use this data to test our APIs and ensure proper access controls are being enforced on our users.

More security testing within our AWS environment will involve role-based access control (RBAC). This will ensure that our frontend users will only be able to access data specific to their users and not escalate privilege. Though we have not made a final decision on data storage, RBAC will make sure that the front-end can only see the correct data. Also, by ensuring that users can't switch roles if they were to access our environment, we can prevent unauthorized edits to our infrastructure.

5.8 Results

Testing our system will ensure that we meet our standards and requirements for functionality and specifications. With each testing variant, we will be able to verify the validity of another part of our system. For unit testing, we will verify that data packets are being sent properly, AWS can read and send data, and that the data can make it to the front-end application. For interface testing, we will focus on ensuring that our IoT sensor network interface is tested together with our AWS infrastructure interface. For integration testing, we will focus on our integration paths of the sensor to base station communication, base station to AWS IoT Core communication, and AWS API gateway to our Flutter front-end application communication. For system testing, we will focus on a more manual form of testing similar to QA to help us find if there are faults in our overarching system that need to be addressed. For regression testing, we will set up a Continuous Integration and Continuous Deployment pipeline to verify that previously written unit and integration tests do not fail when new code is written. For acceptance testing, we will create a document that lists all of our tests with information on whether they were successful and a description of the problem that is being addressed. Finally, for security testing, we have three distinct security areas. The first area will be sensor-to-base station communication, where we will focus on protecting our system against packet sniffing, packet modification, and imposter sensors introduced to our system. The second area will be between our front-end application and AWS infrastructure, where we will mainly focus on penetration testing. The final area we will focus on will be our AWS infrastructure with role-based access control.

6 Implementation

Current implementation of our design includes progress towards round-trip communication between the different layers of our IoT system. We established communication between our Front-End application, AWS Cloud Service, and a mock-basestation written in Python. Due to delayed part orders, we were unable to fully implement a custom LoRaWAN basestation during the Fall 2022 semester. Thus, our first task in the spring will be to fabricate the basestation to enable full round-trip communication. The LoRaWAN Concentrator chip we are waiting for is a Hardware Attached to Top (HAT) design that will facilitate easy setup and low noise with our RaspberryPi. Furthermore, we have manufacturer-provided instructions on how to set up the basestation using the HAT chip in conjunction with our RaspberryPi model. With this in mind, we estimate this to have a quick turnaround time. At the conclusion of the basestation's integration into our IoT system, we will begin security implementation along both communication paths attached to the basestation.

In the meantime, our other subteams will begin implementing our security designs in the cloud and front-end application. These tasks do not rely on having complete round-trip communication implemented. A specific breakdown of these tasks can be found in Section 3.

7 Professional Responsibility

This discussion is with respect to the paper titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

Table 4

Area of Responsibility	**Definition**	**NSPE Canon**
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in areas of their competence; Avoid deceptive acts
Financial Responsibility	Deliver products and services of realizable value and at reasonable costs.	Act for each employer or client as faithful agents or trustees
Communication Honesty	Report work truthfully, without deception, and is understandable to stakeholders	Issue public statements only in an objective and truthful manner; Avoid deceptive acts.
Health, Saftey, Well-Being	Minimize risks to safety, health, and well-being of stakeholders	Hold paramount the safety, health, and welfare of the public.
Property Ownership	Respect property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.
Sustainability	Protect environment and natural resources locally and globally.	

Social Responsibility	Produce products and services that benefit society and communities.	Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.

7.1 Areas of Responsibility

Pick one of IEEE, ACM, or SE code of ethics. Add a column to Table 1 from the paper corresponding to the society-specific code of ethics selected above. State how it addresses each of the areas of seven professional responsibilities in the table. Briefly describe each entry added to the table in your own words. How does the IEEE, ACM, or SE code of ethics differ from the NSPE version for each area?

Comparing to IEEE Computer Society

Work Competence:

NSPE Canon: "Perform services only in areas of their competence; Avoid deceptive acts".

IEEE: Perform services adhering to own level of competence while being honest about that level of competence.

Comparison: They both say the same thing.

Financial Responsibility:

NSPE Canon: "Act for each employer or client as faithful agents or trustees".

IEEE: Do not participate in immoral financial activities. Provide services that do not detriment your employer.

Comparison: They are similar but IEEE focuses on making sure your actions don't hinder the employer.

Communication Honesty:

NSPE Canon:"Issue public statements only in an objective and truthful manner; Avoid deceptive acts.".

IEEE: Be honest and make sure to communicate with affected parties should problems arise.

Comparison: IEEE focuses more on daily conduct while NSPE focuses on organization conduct.

Health, Safety, Well-being:

NSPE Canon: "Hold paramount the safety, health, and welfare of the public".

IEEE: Health, safety and welfare of the public should be the number one focus.

Comparison: They both say the same thing.

Property Ownership:

NSPE Canon: “Act for each employer or client as faithful agents or trustees.”.

IEEE: Provide a fair agreement in terms of ownership.

Comparison: NSPE talks about ownership of property under an organization while IEEE talks about ownership of property in general.

Sustainability:

NSPE Canon:

IEEE: Make sure not to harm the environment.

Comparison: Cannot compare.

Social Responsibility:

NSPE Canon: “Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.”.

IEEE: Identify and report issues of social concern.

Comparison: This differs from the NSPE Canon in the fact that it talks less about perception of the profession and more about what the public expects

7.2 Project Specific Professional Responsibility Areas

Work Competence is an important part of every team, ours included. We currently function at a medium level in relation to work competence. Every team member brings unique expertise to the team that allows us to perform at a high level. However, since we are still young in our careers there is much that we do not know, which we realize and continue to strive to get better every day.

Financial Responsibility is a low level of importance to us. We do not have a budget for our project but we still recognize that we need to be financially responsible with the money we are given. If we are needing to buy a part for our system we spend time making sure we are making a reasonable and correct purchase so as to not waste our client's money.

Communication Honesty, for our group, is a medium level of importance to us. As a team, we have multiple ways to communicate with each other. If one team member is falling behind with their work we communicate with them to figure out the problem and how to go about a solution to the problem.

Health, Safety, and Well-Being are very important for us in a professional context. As a team, we are currently at a low level but plan on focusing more on this area next semester when we start our security implementation.

Property Ownership is a low level of importance to our team. We were given an AWS account by our client to us, and we respect that. We will not use it for any personal project or manage it irresponsibly.

Sustainability is an important aspect for us. We are currently at a high level of performance as a team in this area. At the start of our project, we focused a large portion of our time on making sure our system would be sustainable for farmers so that they can use it from planting season to harvest.

Social Responsibility does not apply to our group. At the moment, we are not developing a system that is important socially speaking.

7.3 Most Applicable Professional Responsibility Area

The most applicable professional responsibility area for this project would be sustainability. Our products will be deployed in fields that require constant maintenance and specific balances of nutrient levels. If our product does not respect this environment and take proactive steps to prevent damage, it will fail as none of our users will want to trade our product for the health of their crops.

We plan on ensuring our systems hold sustainability at the highest standard by regularly testing the durability and composition of our products before deploying them in fields. We only buy and test parts that are rated for outdoor weather conditions and have a long life span. We also research the materials used for these products to ensure there is a minimal amount of possibly toxic materials

8 Closing Material

8.1 Discussion

Thus far, our project has yielded a usable mobile application and simple single-user AWS infrastructure. Users are able to download our application and receive static data from AWS. The only portion of our project that still requires significant development is the IoT sensors & basestations. Moving forward into the next semester, this will be our primary focus, as we hope to have as much time as possible set aside for security and testing. By the end of this year or January of next year, we will have sensors reading and uploading data to our cloud infrastructure. There are a number of functional requirements that still need to be addressed, mainly surrounding the access of live and up-to-date data. However, we will also be dedicating a significant amount of time to developing our intrusion detection systems going forward. This will require additional research into possible machine-learning techniques and other data processing techniques.

8.2 Conclusion

At the time of writing this document, we have completed two different round-trip prototypes. The IoT architecture was mocked out in Python to send data to the cloud, and all services within the cloud have been deployed as a ‘sandbox’ or experimentation version. The prototype is able to decode the mocked sensor value and put that data into a Dynamodb table. Once in the table, the front-end application is able to display the data using a Rest API and a lambda to pull data as a JSON object. The data is then displayed on our Flutter app on the scroll view page in a user-friendly way. In order to meet our goal of easy data visualization we’ve also begun mocking up a graph page for our mobile application that will chart sensor readings over time for the user. We plan on

wrapping up our system functionality at the beginning of the second semester so we can focus the majority of our time on the security of our system.

Our goals for IoT are to provide a secure and constant stream of sensor readings for users. This goal is one we will still be working towards going into next semester. We will accomplish this goal by using the secure and industry standard LoRaWAN OTAA protocol for setup and TCP/IP with MQTT for robust and consistent readings.

Our goals for AWS relate to secure and separated data storage. One major part of this that our efforts will be focused on next semester is the separation of user data. We need to ensure that only one authorized user is able to access their data and nobody elses. This will be accomplished through the use of several different S3 buckets and specialized key pairs to ensure minimal permissions are given to each user.

Our goals for the front end of this project is to provide quick and useful data visualization for our users. This goal is the closest to being completed out of the ones mentioned above. We will need to spend some more time developing graphs and researching useful metrics for farmers. However if we can achieve all of those steps this goal will be completed early next semester.

The main blocker to our progress so far has been figuring out the format we should use to store the data. Additionally, it has taken time for our IoT parts to arrive. If we were able to start this project over, we would have taken more time at the start to look into off-the-shelf sensors. We initially thought that we would be able to use a master's student soil sensor, but that fell through and caused us to fall behind. Regarding the Flutter application, we started to try and integrate our application with AWS's Amplify. This ended up being a dead end for us and caused us to waste around two weeks of our development time.

References

[1]N. Selimović, “ABP vs OTAA,” www.thethingsindustries.com, Nov. 17, 2017. https://www.thethingsindustries.com/docs/devices/abp-vs-otaa/

[2]A. Mishra, “Cloud-based multi-sensor remote data acquisition system for precision agriculture (CSR-DAQ),” Thesis, Iowa State Universiry, 2020.

[3]N. Selimović, “Best Practices,” www.thethingsindustries.com, 2022. https://www.thethingsindustries.com/docs/devices/best-practices/

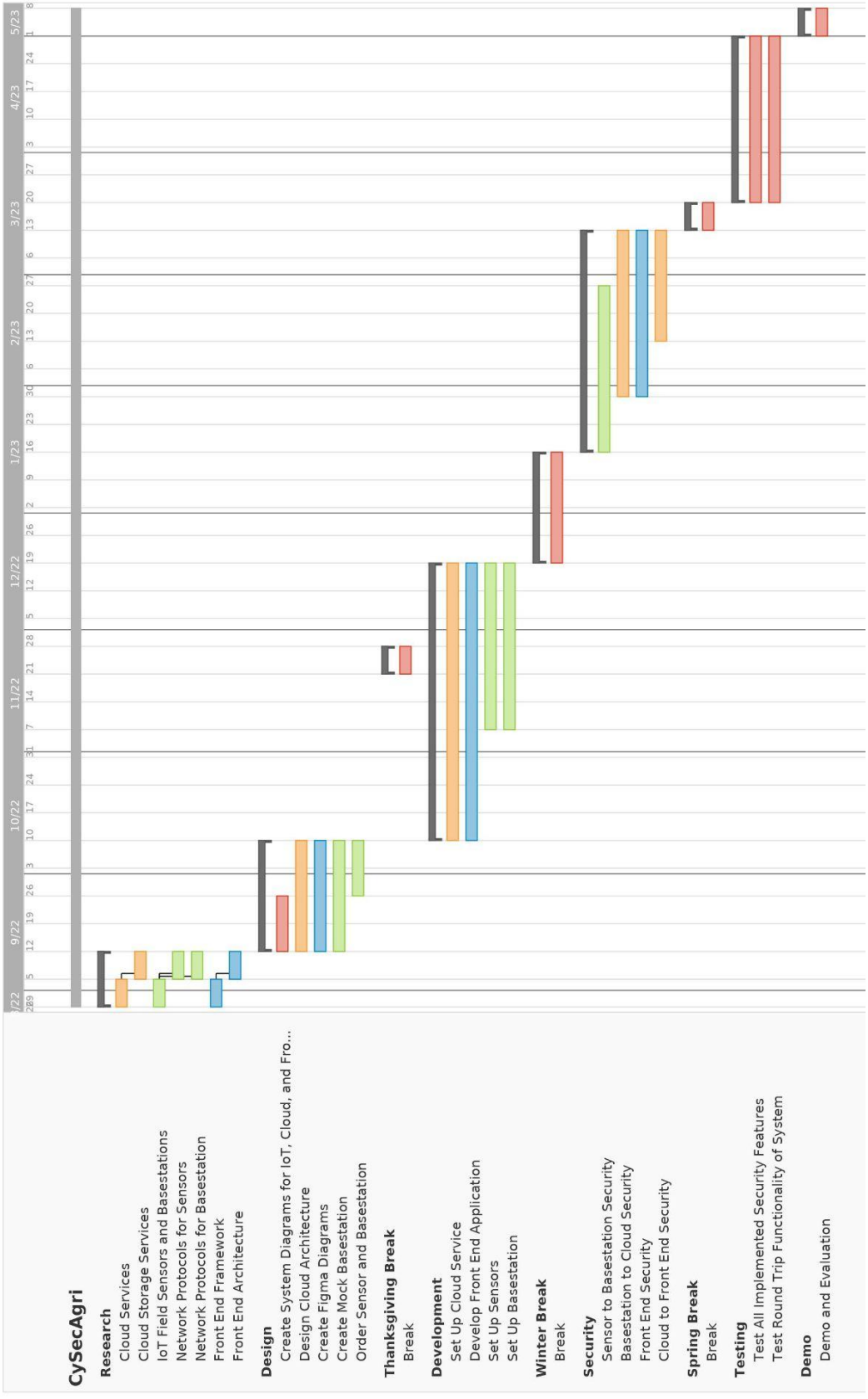
[4]K. Lee and J. Lu, “The New Generation LoRaWAN Sensors of SenseCAP S201X Sensors User Guide.,” Nov. 2022. [Online]. Available: https://files.seeedstudio.com/products/SenseCAP/S210X/SenseCAP%20S210X%20LoRaWAN%20Sensor%20User%20Guide.pdf

[5]M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, “A Survey of Internet of Things (IoT) Authentication Schemes,” Sensors, vol. 19, no. 5, p. 1141, 2019, doi: 10.3390/s19051141.

[6]J. Mccormack et al., “Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment,” International Journal of Engineering Education, vol. 28, no. 2, pp. 416–424, 2012.

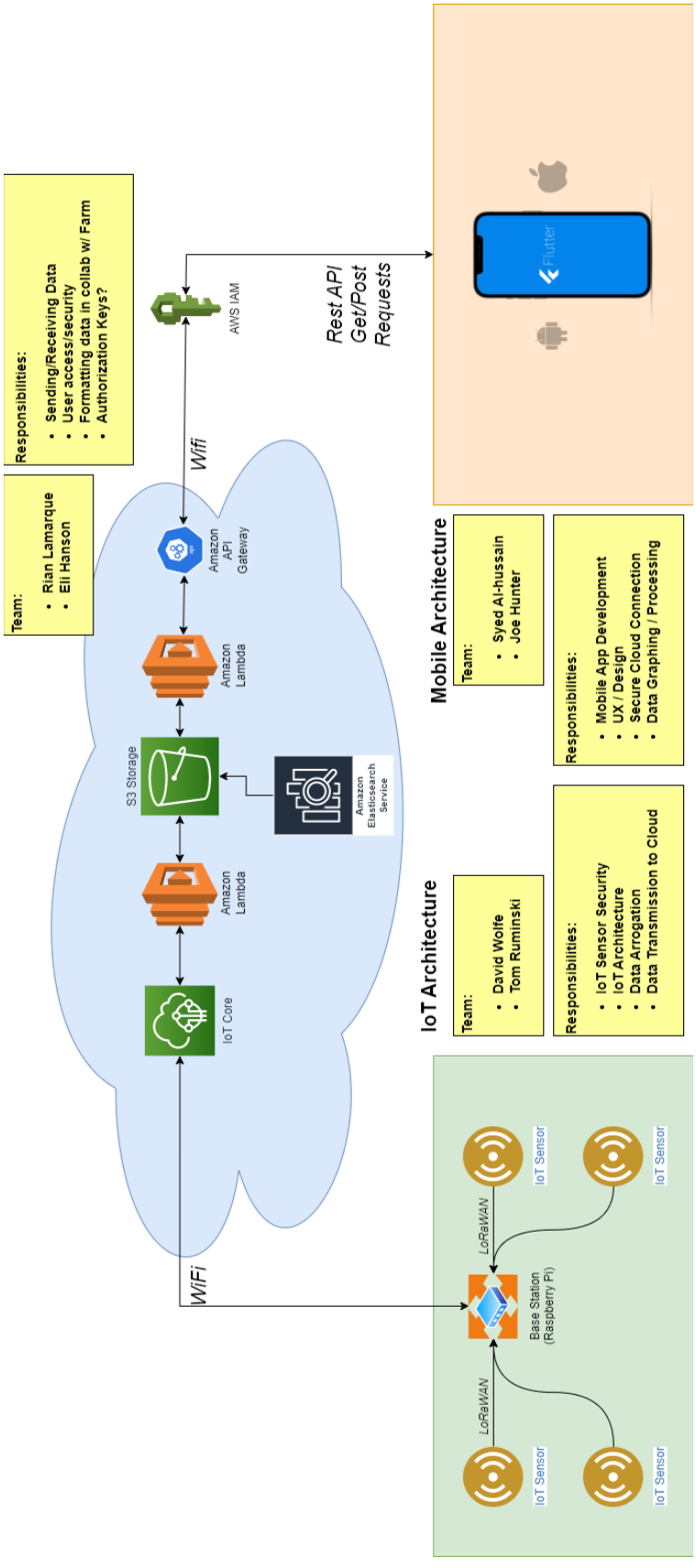
Appendix A

Gantt Chart



Appendix B

System Diagram



Appendix C

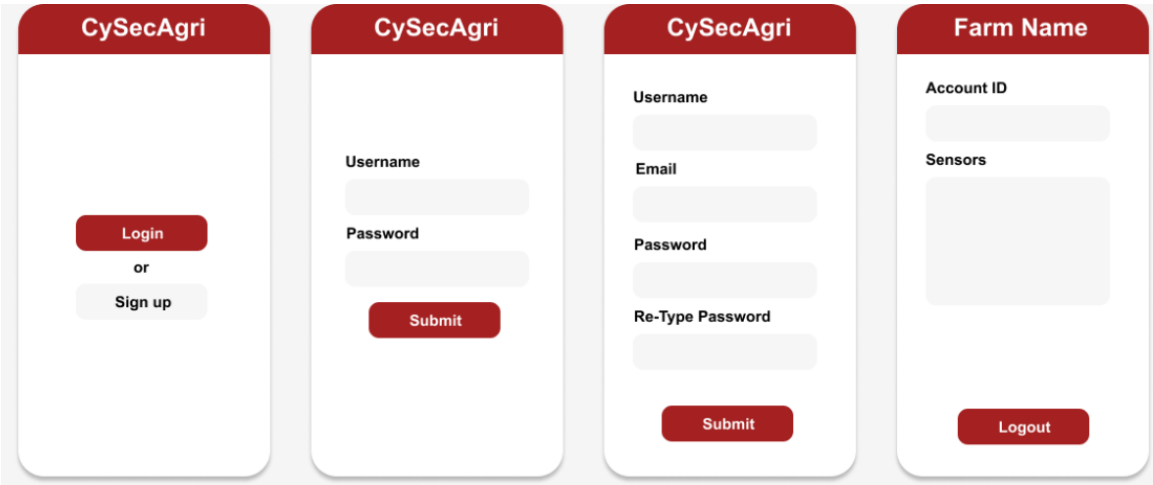
Sensor Data Packet Format

01 0610 245E0000 01 0710 BCB10000 A3D9

Part	Value	Raw Data	Description
1	Soil Temperature	01 0610 245E0000	01 is the channel number. 0610 is 0x1006 (little-endian byte order), which is the measurement ID for soil temperature. 245E0000 is actually 0x00005E24, whose equivalent decimal value is 24100. Divide it by 1000, and you will get the actual measurement value for soil temperature as 24.1℃.
2	Soil Moisture	01 0710 BCB10000	01 is the channel number. 0710 is 0x1007 (little-endian byte order), which is the measurement ID for soil moisture. BCB10000 is actually 0x0000B1BC, whose equivalent decimal value is 45500. Divide it by 1000, and you will get the actual measurement value for soil moisture as 45.5%RH.
3	CRC	A3D9	The CRC verification part.

Appendix D

Figma Page Designs



Appendix E

Risk Severity Table

Appendix F

Team Contract

Team Name CySecAgri

Team Members:
1) David Wolfe 2) Syed Al-Hussain
3) Rian Lamarque 4) Joe Hunter
5) Tom Ruminski 6) Elijah Hanson

Team Procedures

1. We shall meet from 4pm to 5pm every Monday evening on a group Discord call to discuss the weeks events and deadlines.
2. Each member agrees that their preferred method of communication is group or direct messages on Discord.
3. Any issues that are unable to be resolved between the members they affect shall be brought to a group discussion and a majority vote will settle the issue. If a vote is split 50/50 a coin will be tossed to determine the winner.
4. David Wolfe will take and post meeting minutes for both advisor and team meetings in the notes channel of our Discord.

Participation Expectations

1. It is expected that each member attends all team and client meetings.
2. Barring an emergency, if a member must miss a meeting there must be clear communication at least 24 hours prior to the meeting.
3. It is not expected that every member votes in every decision. If a member is indifferent about the results of a decision they may abstain from voting.

Leadership (Roles)

1. David Wolfe - Notetaker and Application Security
2. Joe Hunter - IoT Architecture
3. Rian Lamarque - Diagramming and Cloud Security
4. Eli Hanson - Cloud Security and IoT device Security
5. Tom - IoT Architecture
6. Syed Al-hussain - Application Developer

Collaboration and Inclusion (What skills do you bring to the table)

1. David Wolfe - Pentesting and offensive security experience
2. Joe Hunter - Application development experience

3. Rian Lamarque - Design/Presentation knowledge, AWS Cloud Experience
4. Eli Hanson - Network Architecture and Security experience
5. Tom - IoT concepts, networking basics, security basics, python simulation
6. Syed Al-hussain - ETL pipelines, Web development

Goal-Setting, Planning, and Execution

1. The end goal of our group for this semester is to have a well documented and easy to follow plan laid out for how we will be successful in developing the CySecAgri application in semester 2.
2. Work will be divided based on skill levels and enthusiasm of team members.
3. Tasks will be tracked in threads in appropriate Discord channels.

Consequences for Not Adhering to Team Contract

1. In the event that a task is unable to be completed by the assigned team member, that task will be reassigned to more available team members and the original assignee will have credit taken away for work on that task.
2. Continued missed attendance and/or dropped tasks will result in a discussion between all team members and the course professor.
3. Should any problems continue a meeting should occur to decide how to move forward.

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) _____David Wolfe_____ DATE _____9/20/2022____
2) _____Syed Al-Hussain_____DATE _____9/20/2022____
3) _____Rian Lamarque_____DATE ____9/21/2022_____
4) _____Joe Hunter_____ DATE ____9/21/2022_____
5) _____Tom Ruminski _____ DATE _____9/21/2022___
6) _____Elijah Hanson_____ DATE _____9/22/2022____