Design Context:

Broader Context:

- 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 to perform 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 less sensors and cloud resources that 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

Prior Work:

Mishra, A. (n.d.). Cloud infrastructure for multi-sensor remote data acquisition system for precision agriculture (CSR-DAQ) (thesis).

- 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 sensor to 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 only create a mobile application. We will implement 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.

Technical Complexity:

- At an infrastructure level, our system combines a variety of and 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 raspberry pi 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

Design Exploration:

Design Decisions:

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 less 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 was 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 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.

Ideation:

One decision that needed to be made was: 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.

Decision-Making 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.

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