

AI-Integrated Autonomous Drone System for Smart Crop Health and Harvest Monitoring

sdmay26-20

<https://sdmay26-20.sd.ece.iastate.edu/>

Alex Pallan, Caleb Zea, Z Harvey, Camden Senneff, Srdan Kopunovic, Andres
Lopez

1: Introduction.....	3
1.1. Problem Statement.....	3
1.2. Intended Users.....	4
2: Requirements, Constraints, and Standards.....	4
2.1. Requirements and Constraints.....	4
Functional Requirements.....	4
User Experience Requirements.....	5
UI Requirements.....	5
Database Requirements.....	5
AI/ML Requirements.....	6
Resource Requirements (Constraints).....	6
2.2 Engineering Standards.....	7
1. IEEE 1936.1-2021 – Drone Applications Framework.....	10
2. IEEE 1939.1-2021 – Framework for Structuring Low-Altitude Airspace for UAV Operations.....	10
3. IEEE 14764-2021 – Software Life Cycle Processes: Maintenance.....	11
3: Project Plan.....	11
3.1 Project Management/Tracking Procedures.....	11
3.2 Task Decomposition.....	12
3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria.....	13
3.4 Project Timeline/Schedule.....	15
3.5 Risks and Risk Management/Mitigation.....	16
3.6 Personnel Effort Requirements.....	18
3.7 Other Resource Requirements.....	18
4: Design.....	19
4.1 Design Context.....	19
4.1.1 Broader Context.....	19
4.1.2 Prior Work/Solutions.....	20
4.1.3 Technical Complexity.....	22
Problem Scope and Industry Challenge.....	23
4.2 Design Exploration.....	23
4.2.1 Design Decisions.....	23
4.2.2 Ideation.....	24

4.2.3 Decision-Making and Trade-Off.....	25
4.3 Proposed Design.....	27
4.3.1 Overview.....	27
4.3.2 Detailed Design and Visual(s).....	27
4.3.3 Functionality.....	28
4.3.4 Areas of Concern and Development.....	28
4.4 Technology Considerations.....	29
5: Testing.....	30
5.1 Unit Testing.....	30
5.2 Interface Testing.....	31
5.3 Integration Testing.....	31
5.4 System Testing.....	32
5.6 Acceptance Testing.....	33
5.7 Security Testing.....	34
5.8 User Testing.....	34
5.9 Results.....	35
6 Implementation.....	35
7 Ethics and Professional Responsibility.....	35
7.1 Areas of Professional Responsibility/Codes of Ethics.....	36
7.2 Four Principles.....	37
7.3 Virtues.....	38
8 Closing Material.....	42
8.1 Conclusion.....	42
8.2 References.....	43
9: Team.....	44
9.1 Team Members.....	44
9.2 Required Skill Sets for Your Project.....	44
9.3 Skill Sets covered by the Team.....	45
9.4 Project Management Style Adopted by the team.....	45
9.5 Initial Project Management Roles.....	45
9.6 Team Contract.....	46

1: Introduction

1.1. Problem Statement

The goal of this project is to develop an unmanned drone capable of assisting farm owners in inspecting and evaluating their crops. Our drone will collect data by scanning the fields and then utilize a vision model to compare the gathered data with other information to identify any issues with the crops, such as pests or diseases.

While there may be other companies that have made similar products, we intend to show that it is possible to create a product that can accomplish similar tasks on a cheaper budget. Other companies have created a similar platform, but require farmers to purchase both their own drone as well as a subscription service, in order to use their proprietary app. We intend to create a product that can be created using basic off-the-shelf parts, as well as a desktop application for ease of access.

1.2. Intended Users

The three main user groups for this drone will be farmers, insurance companies, and farming supply companies. Farmers will likely use this drone to monitor their own crops to avoid loss of income and to try and catch potential issues sooner rather than later. Farmers will likely want a user interface that is simple to manage and understand, while still allowing them complete control over the management of the drone and understanding of the information that it provides. Insurance companies may see this drone as a way to avoid/mitigate damages. Since any damage that may happen to a farmer's crops could potentially result in a policy payout for the insurance company, they may seek to utilize our drone to help mitigate risk.

This could also potentially benefit the farmers who use our drone, because if the insurance companies have confirmation that they've been using our technology to monitor their fields, they may receive a better deal on their agriculture insurance policies. One final group that may use our drone is farming supply companies that may seek to privatize this idea. If this idea performs well, different agriculture companies may try and create their own versions of our software for their own private sale. They would be looking for effective, functional work from the drone, and would be looking to see how it performs and how accurately and efficiently the vision model performs when identifying these pests and diseases, as well as seeing how soon we can detect these issues, and if they can be stopped before they can do any real damage. As more companies enter this market, the number of options available to farmers will increase, creating added incentive and benefits for both companies and consumers.

2: Requirements, Constraints, and Standards

2.1. Requirements and Constraints

Functional Requirements

Current:

1. Plant Growth Tracking – Week-by-week imaging to monitor crop development.

Future:

2. Autonomous Drone Flight – Pre-programmed routes with GPS and obstacle avoidance.
3. Insect Detection – AI-based identification of pest infestations.
4. Harvest Time Monitoring – Visual and data-based estimation of crop maturity.
5. Farmer Dashboard – Centralized platform to view growth, pest alerts, and harvest predictions.
6. Reliable predictions of harvest readiness.
7. Cloud Storage – User management and file storage in the cloud, with authentication and best security practices.
8. Local Storage - Data stored locally, to be synced with the cloud database when online

User Experience Requirements

1. Ability for users to set up and configure drone flight through a cross-platform desktop application.
2. User-friendly UI that enables flight planning, field management, and data transfer after the field scan is completed.
3. Generating reports after each field scan for identified crop diseases, as well as crop growth over time.
4. Allows users to save previous field configurations and reuse them
5. Ability to use the application when not connected to the internet, as well as having data sync when a connection is established.

UI Requirements

1. Desktop application. The UI must be simple enough for any farmer to use, especially those unfamiliar with technology.
2. The UI must be robust enough to display unique field sizes and shapes, as well as corn or soybeans, and a scheduled flight time at the farmer's selected intervals. The central dashboard should allow farmers to see all important information quickly.
3. The UI should be aesthetically pleasing with a color theme, while also being bright enough to be visible on a laptop screen in sunlight.

Database Requirements

1. Local storage when offline, to sync when connected
2. Cloud storage for backup and offloading local storage
3. Data to store:
 - a. Flight Data:
 - i. Flight coordinates
 - ii. Flight time
 - iii. Identified hazards
 - iv. Pathing
 - b. Crop data:
 - i. Crop images
 - ii. GPS coordinates
 - iii. Model results (as described under AI/ML Requirements)
 - c. Telemetry data:
 - i. Battery Usage
 - ii. Field Coverage Time

AI/ML Requirements

1. The training dataset includes corn/soybean images across various growth stages, with diverse lighting conditions, stress, and environmental conditions.
2. Each image or image tile in the data set must be labeled according to the standardized annotation guide. The labels include:
 - Current Requirements**
 - a. Growth percentage - predict a growth stage, then map that stage, → percentage
 - Future Requirements**
 - b. Pest presence proxies - a visual sign that suggests insect activity
 - c. Leaf damage severity - ordinal value depends on how severe
 - d. Canopy cover percentage - proportion of the image occupied by green
 - e. Weed presence - binary (weed or no weed) or categorical (broadleaf, grass, none)
 - f. Lodging indicators - evidence of fallen or leaning plants
3. The dataset must be split into train/validation/test sets with no field or time overlap to avoid leakage.
4. Model training must be reproducible with fixed seeds, versioned datasets, and logged hyperparameters.
5. The model should provide explainable outputs (saliency/heatmaps) to highlight regions influencing predictions.
6. All datasets must follow licensing and privacy requirements—used only for research/educational purposes and PII-free.
7. The system should include an active-learning loop to flag low-confidence samples for human review and retraining.

Resource Requirements (Constraints)

1. The drone should cost less than \$750 (at the time of build).
2. The drone should be able to be built with readily available consumer parts.
3. The drone controller software should work with any standard Betaflight drone configuration.
4. Drone design should be able to gather a reasonable amount of data within energy constraints (the average field size of 350 acres on one charge) and support pausing and resuming data collection to recharge the drone for larger fields.

2.2 Engineering Standards

Engineering standards reinforce requirements and safety in creating designs. They are important to adhere to, as they can guide a project when followed, while also cautioning against hazards to keep not only engineers designing it safe, but also users, non-users, and the environment. They also exist to maintain complicity with regulations, morality, and the law.

Four IEEE standards that we found to apply to our project are as follows:

1. IEEE Standard for Drone Applications Framework
IEEE 1936.1-2021
Description: This standard defines all things related to drones and their potential functions. It defines all of the potential systems that drones with different use-cases would have on board, and provides categories for each different type of drone and how they are defined. Our project most likely fits under the “Agriculture” definition for our application scenario.
2. IEEE Standard Adoption of Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) Technical Specification Connected Autonomous Vehicle-Architecture (MPAI-CAV) V1
IEEE/MPAI 3307-2024
Description: This is a standard defining various aspects of an autonomous vehicle that uses AI/ML models to navigate and interpret its environment. It defines the layouts of the sensors, as well as their types. Additionally, it defines different models for how the AI components interact with the other parts of the vehicle, as well as interaction with humans
3. IEEE Standard for a Framework for Structuring Low-Altitude Airspace for Unmanned Aerial Vehicle (UAV) Operations
IEEE 1939.1-2021
Description: This is a standard for structuring the airspace itself for the drone. This is a more zoomed-out view from the above standard (IEEE 1936.1-2021), which is more about the drone itself. This standard defines how an airspace can be interpreted and split into regions or grids, as well as logistics regarding operating the drone. This standard, in particular, is useful because it also covers creating a mosaic of the landscape, which is a component of our project.

4. IEEE Standard for System and Software Verification and Validation
IEEE 1012-2016

Description: This standard defines the processes and documentation needed to verify that a system is built correctly and validate that it meets its intended purpose. It provides structured methods for testing, quality assurance, and performance assessment across all stages of software and system development. This standard is relevant to our project's vision model, as it guides how we verify that our AI pipeline (data preprocessing, training, inference) functions as intended and validate that the model's predictions—such as pest detection, crop growth, and harvest readiness—accurately reflect real-world field conditions. Following this standard ensures systematic testing, measurable performance metrics, and traceable validation evidence for our AI-based crop monitoring system.

After reviewing the technical details of these four standards, this is how they apply specifically to our project:

- 1) The first standard (IEEE 1936.1-2021) directly relates to our design when section five of the standard comments on different drone application types. In our case, it is information collection, transmission, and processing. There are subsections within this that also directly relate to our case, which is using the drone system to obtain target data using artificial intelligence to classify and analyze the data. The operator should also be able to monitor the drone and correct changes in flight.
- 2) The second standard (IEEE/MPAI 3307-2024) also somewhat correlates with our project, as it discusses the interaction of humans with connected autonomous vehicles. The standard states in section 5.1 that a CAV is defined as a couple of functions that our project will include. The CAV must converse with humans; in our case, a flight path must be taken based on user specifications. The standard also includes that the CAV must plan a route to reach the requested destination, and it will autonomously reach it by taking a few logical steps that are not included in our design specifications, such as building a complete digital representation of the environment.
- 3) The third standard (IEEE 1939.1-2021) is extremely relevant to our design in that it covers standards for unmanned aerial vehicles (UAVs) in low altitude airspace, under a few hundred meters, which is the space our drone will be operating in. It covers the specific data structure and notation that should be used by our drone when subdividing the area where data collection is occurring. It also notes several requirements for how closely the drone should be able to fly within the designated flight plan (for example, altitude must be within 10% of the designed altitude, route curvature no greater than 3%, rate of ascent no more than 10m/s), in addition to safety requirements before the launch of a UAV. These requirements are designed to ensure that the data collected is as reasonably accurate as it can be, which is critical for the level of precision our drone will require for useful results.

- 4) The fourth standard (IEEE 1012-2016) directly relates to our project's vision model and AI data processing pipeline. It establishes a structured framework for verifying that our software components—such as data preprocessing, image classification, and AI inference—are implemented correctly, and for validating that the model's outputs align with real-world crop conditions. This connection is critical since our system's accuracy depends on how well the AI model identifies pest presence, crop health, and harvest readiness based on field imagery. Following this standard ensures that our model undergoes systematic testing, meets performance thresholds, and provides traceable validation evidence that supports its reliability in real agricultural environments.

After discussing with other members of the group, here are some other standards we found that apply to our project:

1. IEEE International Standard - Software engineering - Software life cycle processes - Maintenance IEEE 14764-2022
Description: This is a standard describing how to perform the iterative process of development for software. The standard can be applied to any software product, and especially takes into account the maintenance of software into account. The criteria outlined in this standard include planning and maintenance of software, as well as the execution of software maintenance activities. It aims to outline how software maintenance should be planned during software development, and then provides criteria to assess the effectiveness of software maintenance execution. The standard doesn't address the operation of software, backups, recovery, or system administration.
2. IEEE Standard for Information Technology– Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications IEEE 802.11-2024
Description: This long-titled standard defines how wireless local area networks operate over short distances. This standard affects how devices like laptops, phones, and routers communicate with each other while on Wi-Fi. The definitions in this standard include the Medium Access Control or MAC layer, which controls how multiple devices share a radio channel without interference with each other. The Physical Layer specifies how bits are transmitted in the air with the help of radio waves. Finally, the standard ensures that interoperability is maintained, in other words, the ability for devices from different manufacturers to be able to communicate with each other reliably. This standard is the foundation for Wi-Fi, addressing how data physically moves wirelessly and how devices work together while on the same Wi-Fi.
3. IEEE Standard for Software Engineering – Software Life Cycle Processes – Maintenance IEEE 14764-2022
Description: This standard defines the processes and best practices for maintaining software throughout its life cycle. It focuses on planning, managing, and executing maintenance activities after the initial development phase, ensuring that software continues to function correctly and efficiently over time. The criteria outlined include documentation, modification control, error correction, and performance improvements. This standard applies to our project because the AI vision model and drone control software will require regular updates—such as retraining models with new crop data, fixing bugs, and optimizing performance—to maintain accuracy and reliability as environmental conditions and datasets evolve.

Some modifications we plan to make to incorporate these standards apply as follows:

1. IEEE 1936.1-2021 – Drone Applications Framework

This standard defines categories of drone systems and outlines the onboard subsystems required for different applications. To maintain this standard, we will:

- **Categorize our drone under agricultural operations**, following IEEE classification, and define subsystems as navigation, imaging, and environmental sensing.
- **Map system architecture** (hardware components, AI components, and communication modules) to the standard's modular framework.
- **Document system interactions** using standardized terminology from IEEE 1936.1, improving clarity for future integration with other packages.

2. IEEE 1939.1-2021 – Framework for Structuring Low-Altitude Airspace for UAV Operations

This standard provides a structured model for UAV airspace management and coordination. To maintain this standard, we plan to:

- **Design the drone's flight control logic** to comply with airspace segmentation and altitude bands, ensuring safe autonomous operation in low-altitude regions.
- **Integrate a mapping mode** consistent with this framework to standardize how imagery and field data are collected and aligned to geospatial grids.

3. IEEE 14764-2021 – Software Life Cycle Processes: Maintenance

This standard defines the framework for planning and executing software maintenance. To maintain this standard, we will:

- **Establish a formal maintenance plan** early in development, detailing packaging, architecture, and testing.
- **Use version control and CI/CD pipelines** to track maintenance activities and regression tests.
- **Adopt iterative review cycles** for updating both flight software and the AI processing microservice, ensuring reliability.

4. IEEE 1012-2016 – System and Software Verification and Validation

This standard establishes guidelines for verifying and validating software systems to ensure they function as intended and meet user requirements. To incorporate this standard, we will:

- Develop a structured **V&V plan** detailing verification steps (unit and integration tests) and validation procedures (accuracy benchmarking and field evaluation).
- Maintain **traceability** between requirements, dataset specifications, model configurations, and validation results.
- Use standardized **performance metrics** (e.g., precision, recall, and inference latency) as acceptance criteria for validation.
- Document all verification and validation evidence—including test results, error analyses, and retraining outcomes—to ensure transparency and compliance with IEEE 1012 guidelines.

3: Project Plan

3.1 Project Management/Tracking Procedures

The management style we plan to use is agile, as there are many moving parts to consider, including the frontend and backend of our project, as well as the AI component. Agile allows us to work on multiple parts of the project at the same time, while having a group goal to ensure that we stay on track to integrate those parts together.

Progress will be tracked via pushes and merge requests in the group Git repository.

Communication will be made in the group Discord server, with separate channels for different sections of the project, so as not to clutter team channels with unnecessary information.

This method of communication will help the group to understand what each individual is working on, as well as understand how each part contributes to the whole.

3.2 Task Decomposition

Software task decomposition, in ideal linear order

1. Set up frontend and backend environments and get all versions set up, making sure that everything is compatible with each other
 - a. Sync up all versions within Git
 - b. Ensure that our environment works with CI/CD
 - c. Check for incompatibilities between Linux/Mac/Windows/Android libraries
2. Split up work and start development on frontend and backend sections.
 - a. Frontend begins working on desktop application and UI
 - b. Backend develops database and start working on AI implementation, even if it's minimal
 - c. Make sure there is connectivity between frontend, backend with REST, and AI program by syncing and testing on git.
 - i. Information should be able to come from the backend where AI information will be stored and displayed on the UI.
 - d. Can set up CI/CD tests to make future testing and deployment easier.
3. Continue developing on all ends of the software
 - a. Frontend fully develops the UI for the desktop application, ensuring critical requirements and functions are implemented. The farmer dashboard should be accessible at this point.
 - i. Develop UI that allows farmers to see flight information, drone information, and crop information, integrate Google Maps API
 - ii. Ensure that information from the AI microservice is displayed and updated on the UI
 - iii. Program the application to produce a report after every flight to give a summary to the farmer

- b. Cloud and local storage should be implemented at this point.
 - i. This can be performed by anyone on the software team
 - ii. Store frontend UI information locally so that when the application isn't connected to Wi-Fi it will display properly and sync when re-connected to wi-fi
 - iii. Connect the database to make sure it works with local and cloud storage as well.
 - c. The AI should be given its dataset to learn from, and the microservice should be functional. After AI has processed the dataset and starts producing predictions, the frontend should be ready to receive that data and display it to the UI for plant growth tracking, insect detection, and harvest readiness.
 - i. AI should be able to predict: Pest presence, leaf damage, canopy cover, weed presence, and lodging indications.
 - d. Database development should be finalized at this point, supporting all of the information that needs to be stored.
 - i. Flight Data: flight coordinates, flight time, hazards, and pathing
 - ii. Crop Data: crop images, GPS coordinates, model predictions
 - iii. Telemetry data: battery usage, field coverage, time date information
- 4. Integrate all pieces of software together and ensure that they all work together
 - a. At this point, there should be full flights and data exchange that syncs with the cloud while on Wi-Fi.
 - b. The desktop application should be fully functional at this point, allowing the user to view any information about flights and crops.
 - c. Document all of our work

Since we're an agile team, each of the numbers relates to where the sprints should be spent

- 1. Sprints 1-2
- 2. Sprints 2-5
- 3. Sprints 5-13, biggest chunk of our project is here
- 4. Sprints 13-16

Hardware task decomposition

- 1. Order parts for the drone after doing extensive research on different possibilities to accomplish our goals
- 2. Assemble the drone physically, ensuring that it can handle being flown for hours a day. Durability is an important aspect.
- 3. Connect the drone with open-source flight APIs that allow the drone to be flown, keeping GPS coordinates and flight information on board until reconnected with the charging station.
 - a. Data must be transferred between drone and the desktop application while reconnected
- 4. Test Test Test, the drone should be able to avoid obstacles, do unique flight paths and field shapes, and not run out of battery while in the air.
- 5. At this point, the drone should be fully functional, durable, consistent, and interconnected with all of the software without issues

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

I. Overview

The goal of our project is to develop an Autonomous Drone with the capability to monitor and analyze crop fields for agricultural producers. This includes several components:

- Custom drone with the ability to fulfill all functional requirements specified in the initial project documentation, as well as a
- Cross-platform desktop application
- REST API for data management
- Microservice for AI processing

II. Major milestones

Milestone	Description	Delivareable	Target
1. Init and Setup	Establish Git repo, define architecture, set up development environments	Version control repo	Week 4
2. Hardware Integration	Assemble drone, integrate flight controller, GPS, camera, and sensors	Complete drone with flight, camera, and sensor functionality	Week 9
3. Backend & API	Implement REST API for data storage, flight logs, and telemetry data	Functional API with at least 60% functionality	Week 10
4. AI Integration	Develop an image-processing and crop-analysis pipeline; train an initial model using sample data	AI microservice with at least 70% accuracy	Week 11
5. Desktop App	Create cross-platform UI for control, flight monitoring, and data visualization	Desktop app for flight visualization and API integration	Week 12
6. Testing	Integrate drone, backend, and AI microservice with the desktop app	Full flight and data exchange	Week 13
7. Optimization	improve ML model performance, enhance UI/UX Improve growth-stage accuracy	Improvements on identified issues	Week 14
8. Documentation	Create documentation for each component	Component and instruction documentation	Week 15

III. Metrics & Progress

Drone

- Autonomous flight is enough to cover the field
- Accurate flight based on path determined (≤ 3 m error)
- Sufficient data collected (Full GPS data, sensor readings, and imaging)

Database & API

- 100% of defined REST endpoints are functional
- Unit test coverage $\geq 70\%$

AI Microservice

- Initial accuracy target 70%
- Dataset size ≥ 2500

Desktop Application

- All primary UI elements implemented
- Data visualization
- 0 critical UI/UX bugs

Team Metrics

- At least one deliverable per sprint
- $\geq 80\%$ Git issues closed per sprint
- Code reviews completed within 72 hours

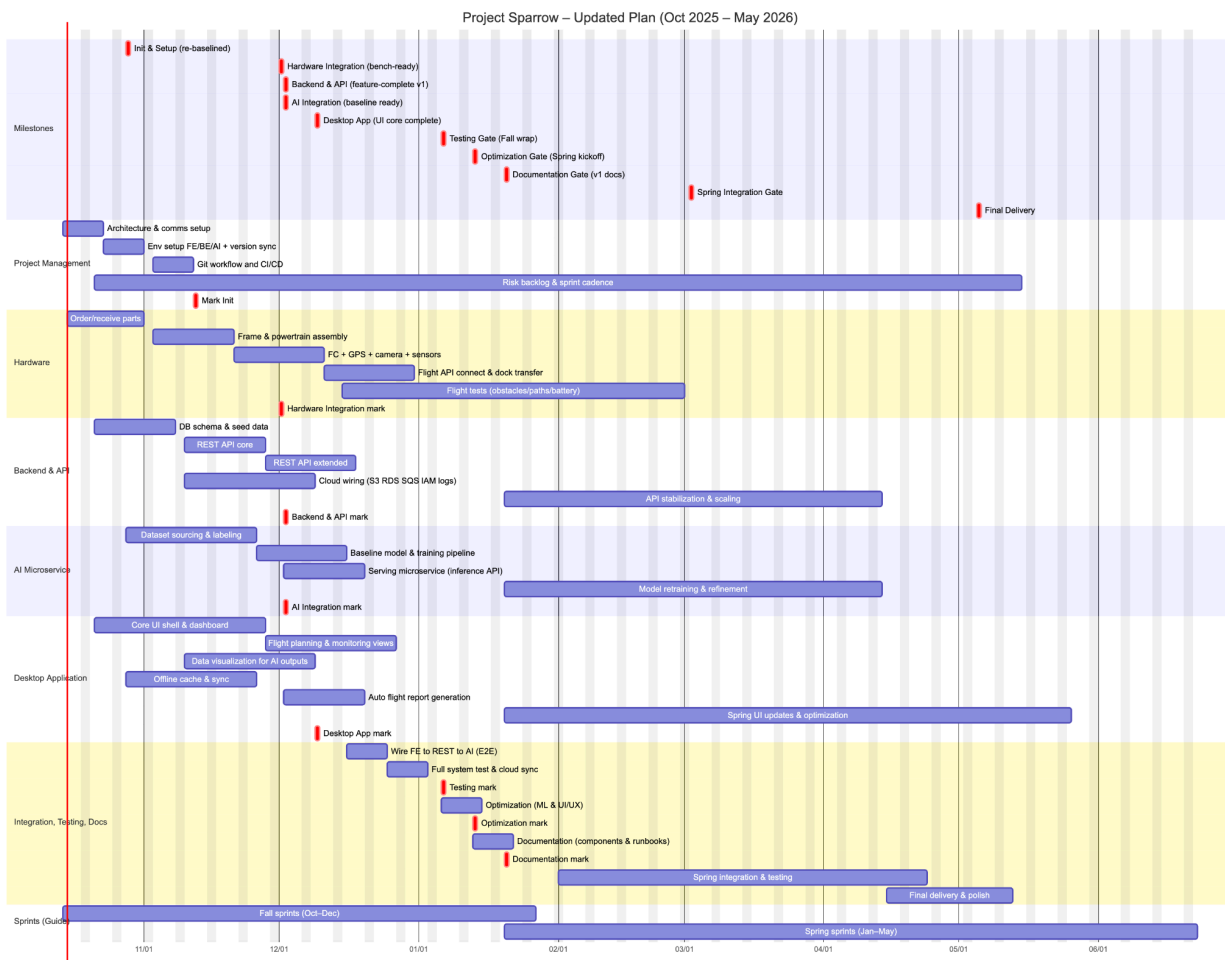
IV. Evaluation Criteria

- Drone achieves autonomous navigation across a predefined area with $> 90\%$ completion success.
- AI model meets target accuracy ($\geq 70\%$) requirements.
- Backend/API passes testing and satisfies all data requests
- Data visualization and upload success rate $\geq 90\%$.
- System architecture maintains modularity and scalability for future integration
- Code quality and documentation meet basic review standards

V. Iteration and refinement

As the project progresses, goals and standards will be updated to meet realistic expectations and help improve the team's development cycle. Future iterations may refine AI accuracy requirements, drone field coverage success rate, API and database performance, and UI/UX design goals.

3.4 Project Timeline/Schedule



3.5 Risks and Risk Management/Mitigation

Our project involves integrating multiple complex systems, including drone hardware, onboard AI vision models, cloud connectivity, and a farmer dashboard. As a result, several technical and operational risks have been identified across our major tasks. Each risk is evaluated based on its probability and severity, with mitigation strategies proposed for those exceeding a probability of 0.5 or those with a high potential impact.

Agile Risk Tracking
Risks will be re-evaluated during each sprint.

Task	Risk Description	Probability	Severity	Mitigation / Management Strategy
Drone Flight and Navigation System	GPS-based navigation may experience signal loss or instability in rural areas.	0.6	High	Integrate fallback IMU-based navigation and manual override mode. Consider using an RTK-GPS module or off-the-shelf autopilot (e.g., Pixhawk) for improved precision.
	Limited battery capacity may prevent reaching the 45-minute flight target.	0.7	High	Investigate higher-capacity Li-ion packs or lighter frame materials. Schedule test flights to optimize power draw by adjusting motor speed and sensor sampling frequency.
Vision AI Model (Plant Growth, Pest Detection, Harvest Estimation)	Model accuracy may be insufficient due to lack of locally collected crop data.	0.8	High	Request dataset access from agricultural research institutions in Iowa (e.g., ISU Ag Science Dept., Bayer, Corteva). In the short term, train a general-purpose vision model on public datasets (PlantVillage, CropDeep).
	Training infrastructure may be too slow for iterative retraining.	0.5	Medium	Use AWS EC2 GPU instances for training; enable model checkpointing for partial training progress recovery.
System Integration (.NET ↔ Python Microservice ↔ AWS)	Communication between the .NET backend and the Python microservice could fail due to API incompatibility or latency.	0.6	High	Define clear API contracts (JSON schema). Add logging and retry mechanisms. If persistent issues arise, containerize both services in a shared environment (e.g., Docker Compose).
	AWS service limits or misconfiguration could lead to data loss or downtime.	0.4	High	Implement S3 versioning and CloudWatch monitoring. Use IAM least-privilege roles and periodic backup snapshots.
Farmer Dashboard and Data Visualization	UI performance may degrade with large datasets or real-time feeds.	0.5	Medium	Use incremental rendering techniques and pagination. Offload analytics to AWS Lambda or preprocessed summaries.
Hardware / Sensor Integration	Camera or environmental sensor modules may not be compatible with chosen drone control board.	0.6	High	Verify interface compatibility (I2C/SPI/USB) early. If incompatibility persists, consider purchasing off-the-shelf compatible sensor suite.

Sprint 1–2: Emphasis on hardware compatibility and base flight control.

Sprint 3–4: Vision model dataset acquisition and preliminary model training.

Sprint 5–6: Integration testing and cloud synchronization.

Each sprint will maintain a “risk backlog,” where mitigations (such as fallback algorithms, component replacements, or cloud service adjustments) are logged and prioritized for review during sprint retrospectives.

3.6 Personnel Effort Requirements

Task	Time (hrs)
Hardware Integration Assemble drone, integrate flight controller, GPS, camera, and sensors	10
Backend & API Implement REST API for data storage, flight logs, and telemetry data	60
AI Integration Develop image-processing and crop-analysis pipeline; train initial model using sample data	60
Desktop App Create cross-platform UI for control, flight monitoring, and data visualization	50
Testing Integrate drone, backend, and AI microservice with the desktop app	100
Optimization improve ML model performance, enhance UI/UX	40
Documentation Create documentation for each component	40

3.7 Other Resource Requirements

Hardware Requirements

The drone used in this project is custom-built using components readily available online, with an approximate total cost of \$600. Our suggested drone build is as follows:

- [Flywoo Explorer LR 4 Drone](#) (\$192)
- [3000mAh 4S Li-Ion Batteries](#) (\$90)
- [Raspberry Pi Zero 2 W](#) (\$31)
- [Raspberry Pi HQ Camera](#) (\$71)
- [Remote ID module](#) (\$35)

Additionally, the following parts are needed to interface with and maintain the drone:

- [4S LiPo battery charger](#) (\$35)
- [ELRS RC controller](#) (\$73)
- [18650 battery](#) (\$15)

Cloud Requirements

Our project will be performing data analysis in a backend server hosted within a cloud provider. We have chosen AWS to provide this functionality. Our proposed cloud architecture includes:

- **AWS Lambda** or **EC2**: to host the C# backend that handles the data the drone produces
- **AWS SQS**: a message queue to handle the images uploaded by the drone asynchronously
- **AWS EC2**: to perform AI processing on the images in the SQS queue and publish the results to our database
- **AWS RDS** (Postgres): to store the results of data processing, drone telemetry information, and authentication data
- **AWS S3**: to store the raw images uploaded to the backend from the drone

User Hardware Requirements

To operate the drone in the field, one of the following desktop platforms is required (to operate the client application, which will create a flight plan and upload it to the drone prior to takeoff):

- macOS >= 14
- Windows >= 10
- Debian >= 9 / Ubuntu >= 16.04 / Fedora >= 30

4: Design

4.1 Design Context

4.1.1 Broader Context

Area	Description	Examples
Public health, safety, and welfare	Farmers will have the ability to purchase our drone platform and fly it over their fields for a report on the status of their fields. Within communities, equipment is often shared between farms so one drone could be used for multiple farms.	Decreased cost of field monitoring tools, earlier identification of disease or pests, finding areas of improvement for future years (dead spots, pests), increased overall yield over time
Global, cultural, and social	Farmers are often opportunistic, and so providing a cost effective tool to be an early warning or crop monitoring system is a value that our cultural group holds. Farmers are not afraid to adopt cutting edge technology, and this is just one more example of a new technology they are adapting to improve their operation.	Some communities view drones negatively citing privacy concerns, e-waste from another device, laws could be a limiting factor in some areas.
Environmental	Having a better understanding of what is affecting crops and how to treat it is the core environmental effect we are having with this project. More informed decisions can be made regarding what chemicals to use or solutions to implement. This reduces unnecessary negative environmental actions from agriculture.	Decreasing usage of chemicals, more efficient usage of resources, remediation of issues earlier on, better awareness of problems, more precise field management.
Economic	Our project aims to provide a cost effective solution to field monitoring systems compared to competitors. This reduces the barrier to entry to more farmers for similar tools. Through the usage of our tool, more proactive measures can be taken to fix issues, and resources can be used in a more efficient manner, reducing costs.	Competitive pricing, identifies problems earlier, identifies crop yield problems, farmers don't have to waste money and time on implementing the wrong solution, and increases efficiency for overall operations

4.1.2 Prior Work/Solutions

There exist a ton of similar options in the market. There are almost a dozen established businesses that use AI to analyze a multitude of things about crops. Similar to our project, these companies have developed an AI, trained it on a dataset, and now sell their product for farmers to buy and use to get the most information about their crops.

From what we can tell, using AI in this capacity is new, with all available studies and articles coming out in 2025. This is a growing industry as AI capability continues to advance. One of the core reasons why these businesses are popping up is because the AI works extremely well. “Achieved exceptional predictive performance, with R^2 scores reaching 0.92, mean squared errors as low as 0.02, and mean absolute errors of 0.015 [1].” This quote shows that the AI is highly accurate, and has low error rates. While most businesses don’t use drones (we will touch on this later), one of the studies mentions how “drones can be used in locations for real-time monitoring by capturing aerial data to access crop health and detect diseases [2].” This excerpt sounds eerily similar to our project and its goals. This paper was published in April this year, and outlines some of the strengths and existing challenges to making this idea a reality. It’s a great tool for us to use as we continue to develop our software.

We know AI is good at its job and that there is at least one paper that outlines our entire project, so let’s look at the challenges of using AI. “High upfront costs make AI technology inaccessible for many small-scale farmers, limiting adoption. Technical expertise is another barrier, as many farmers struggle to integrate AI. Reluctance to share farm data hinders AI model improvements [3].” While we can’t directly relieve some of these challenges, namely price inaccessibility, we can definitely produce UI that is friendly to farmers and allows people of all technological backgrounds to use and benefit from AI analysis. After researching the products that businesses have to offer they are much more robust than ours. They do many different types of crops, tell you when to plant, canopy-moisture analysis, when to harvest, and many more data points of analysis.

The way that our project is different from all of the businesses that we could find is the drone. The businesses all use satellite imagery to take pictures of fields and then have their AI analyze those pictures, they don’t use drones. This is where our project has a unique attribute, which is a blessing and a curse. None of the AI datasets are drone pictures, they’re all satellite images, so we can’t train our AI on satellite images and then have our drone take the pictures. Given the similarity of our project with existing businesses, the differences are pretty clear. The pro of our project is that we’re programming and using a drone to take the pictures, which could allow for better data as it’s much closer and should get a better view of the crops, but at the very least it’s a unique approach to a current innovation with AI. The biggest con of our project is the scope, we’re only focusing on corn and soybeans, and we simply won’t have as many features as these fully fleshed out businesses. We are not following any previous work.

Electronic Documents

Professional Internet Site

- [1] R. N. V. J. Mohan, P. S. Rayanoothala, and R. P. Sree, "Next-gen agriculture: integrating AI and XAI for precision crop yield predictions," *Frontiers in Plant Science*, vol. 15, Jan. 2025. [Online]. Available: <https://doi.org/10.3389/fpls.2024.1451607>. [Accessed Oct. 28, 2025]
- [2] N. Aijaz, H. Lan, T. Raza, M. Yaqub, R. Iqbal, and M. S. Pathan, "Artificial Intelligence in Agriculture: Advancing Crop Productivity and Sustainability," *Journal of Agriculture and Food Research*, vol. 20, p. 101762, April. 2025. [Online]. Available: <https://doi.org/10.1016/j.jafr.2025.101762>. [Accessed Oct. 28, 2025]

General Internet Site

- [3] Dr. Chantal Spleiss, "AI in Agriculture: Smarter Crops, Better Yields | CSA," *Cloudsecurityalliance.org*, Feb. 2025. [Online]. Available: <https://cloudsecurityalliance.org/blog/2025/02/10/ai-in-agriculture-smarter-crops-healthier-livestock-better-yields#>. [Accessed Oct. 28, 2025]

4.1.3 Technical Complexity

Our project demonstrates high technical complexity through its integration of multiple hardware and software subsystems, each employing distinct scientific, mathematical, and engineering principles. The overall goal is to develop an AI-integrated autonomous drone system for crop health monitoring, which requires interdisciplinary expertise across control systems, embedded hardware, artificial intelligence, and cloud-based software design.

Flight control and Navigation

This subsystem applies control theory, kinematics, and embedded systems engineering to achieve stable and autonomous flight. It relies on GPS and IMU data for navigation, using PID controllers and sensor fusion algorithms (integrating accelerometer, gyroscope, and magnetometer data) to maintain orientation. Firmware configuration through ArduPilot/Betaflight, combined with communication via DroneKit-Python or equivalent SDKs, enables precise waypoint navigation and telemetry tracking.

Imaging and AI Vision

The imaging module combines computer vision, machine learning, and signal processing to analyze aerial data. A high-resolution onboard camera collects images of crop fields, which are processed by convolutional neural networks (CNNs) to detect signs of disease, nutrient deficiency, or water stress. First iteration focuses exclusively on growth-stage detection. Additional indicators will be added once sufficient dataset and integration maturity are reached. The underlying mathematical principles involve convolution operations, feature extraction, and supervised classification. Training and inference are performed using PyTorch or TensorFlow, with results validated against real-world agricultural datasets.

Backend Processing and Cloud Integration

The backend handles data management and distributed computing. Collected images and telemetry data are transmitted to the cloud using RESTful APIs built with .NET and Python microservices. The system performs data normalization, aggregation, and geospatial mapping before storing results in cloud databases (e.g., AWS). This requires knowledge of database architecture, network communication protocols, and statistical data modeling.

Frontend Dashboard and Visualization

Built using Electron + Vite + React, the frontend integrates software design, data visualization, and human-computer interaction principles. The dashboard displays crop health maps, flight logs, and AI predictions in an intuitive layout. Communication is synced with the backend when drone is charging/plugged in, which allows users to view drone telemetry and processed imagery. Interface design follows IEEE 1936.1-2021 (Drone Applications Framework) for standardized subsystem communication and IEEE 14764-2022 for software life-cycle processes.

Power and Hardware Optimization

Achieving the targeted 45-minute flight time requires the application of electrical and mechanical engineering principles. The team uses energy modeling equations to balance thrust, weight, and power consumption, optimizing motor-propeller efficiency and ESC performance. These calculations determine the appropriate battery capacity and hardware layout to ensure stable flight under variable conditions.

Problem Scope and Industry Challenge

The scope of this project matches current industry standards by integrating AI analysis, autonomous navigation, and cloud synchronization into a unified platform. Unlike most commercial AI agriculture systems, which rely on satellite imagery, our system utilizes drone-based data collection, offering higher-resolution imagery and more adaptable field coverage.

Key technical challenges include:

- Maintaining autonomous stability and accuracy under environmental variability (wind, lighting, GPS noise).
- Designing robust cloud pipelines capable of synchronizing multiple data sources efficiently.
- Extending battery life and reducing payload weight while supporting high-performance sensors.
- Training our own AI model on unique, locally sourced images taken from a drone of corn and soybeans to provide analysis on the crops.

Together, these elements demonstrate that our project is not only technically complex but also ambitious in scope, integrating multiple scientific and engineering disciplines to deliver a system that approaches the sophistication of current commercial agricultural drone solutions.

4.2 Design Exploration

4.2.1 Design Decisions

A major design decision we had to make for this project was selecting the appropriate drone to use. We had a number of options available ranging from pre-built hobbyist drones, to partially built drone kits, to assembling a drone from a completely custom parts list that suits our needs. There were numerous tradeoffs with each approach. A pre-built drone would give us the highest probability of having a working drone, but would have restricted us severely in terms of the hardware customizability (to add an on-board computer, cameras, and other sensors), and in terms of the software SDKs available in that we would be completely limited to whatever APIs the drone manufacturers exposed to us. On the other hand, a custom drone provides us with complete control over the software and hardware, but has a lower chance of success in that we may encounter hardware incompatibilities during assembly and would have to do more work software-wise to get the drone flying. This decision was further complicated by the unavailability of most drones and drone parts due to tariffs. Ultimately, we landed somewhere in-between the two, creating a semi-custom drone build using specialized drone parts following the specifications from one of their recommended drone builds. This way, we know the hardware will be compatible, but we also have the ability to implement any additional hardware or software we need.

Another key decision we had to make was what framework to use for our frontend design. Initially, we were between .NET MAUI and Electron due to the need for cross-platform compatibility (we wanted our app to be accessible regardless of computer OS), and decided on .NET via a team vote, because it offered more support for cross-platform compatibility with added support for Android/iOS if we decided to compile for a phone application. We later revisited this decision when we discovered that .NET MAUI does not compile for Linux and would thus be unable to communicate or compile with our CI/CD testing pipeline in the Git repository. We decided instead to compile with Electron-Vite.

The final design decision focused on planning the backend architecture to integrate with the AI microservice. To ensure stability, maintainability, the team chose to implement the backend using ASP.NET Core, with Entity Framework Core (EF Core) managing the data access layer.

The backend's API layer serves as the communication bridge between the user interface, database, AI microservice, and cloud infrastructure. This modular structure supports parallel task execution and enables the implementation of fallback systems to maintain reliability in case of component failures. All core operational logic is encapsulated within the API layer, which leverages both server resources and AWS cloud services to ensure system performance.

4.2.2 Ideation

We decided on the drone hardware to use for this project by finding any viable options for the project that could meet the minimum requirements (in terms of fly time, software control, and camera and GPS availability), then created complete parts lists for each drone in a spreadsheet. We then evaluated the pros and cons of each drone option in these categories:

- Cost. Is the drone under (or near) the \$500 budget we'd like to target?
- Work to fly. How much will we have to do before the drone will be ready to fly?
- Software customizability. Will we be able to easily implement the autopilot software that we need in order to collect data?
- Hardware customizability. If the required sensors are not on the drone itself (or inaccessible), will we be able to add them ourselves?
- Risk. Is it likely that we will encounter some issue that prevents us from getting the drone in the air? Additionally, will we be able to implement all of the software that we need?
- Flight time. Will we be able to collect a reasonable amount of data before the drone needs recharged? For custom builds, we estimated this using eCalc.

Ultimately, we had 5 real contenders for the drone after a few revisions:

- [DJI Mini 3](#). This drone is completely pre-built and we would be able to get one quickly. It has a very strong fly time and would be easy to get flying, but limits us on custom hardware and software with the SDKs made available. We originally chose a DJI Mini Pro 4 for this, but those are no longer available.
- [Custom 10" Build](#). This class of drone can carry a lot of weight, meaning we would get near the best fly time we could get for a custom drone within our budget. We also get complete hardware and software control, but it is the most risky choice.
- [Custom 7" Build](#). The smaller size means we could use smaller motors, props, and batteries, which improves the cost. However, we lose some flight time. Same risk as the 10" build.
- [Partial Custom 4" Build](#). This drone is built mimicking a pre-built drone provided by the same supplier, nearly guaranteeing hardware compatibility and giving us complete hardware and software control. It gives us a good flight time, but less than all the other options. Part availability is questionable.
- [Pre-Built 4"](#). This is the drone the custom 4" build mimics. We get less hardware customizability and less control over how the parts are assembled. The flight controller runs open-source software that provides a UART control interface for our on-board computer.

4.2.3 Decision-Making and Trade-Off

To decide on the drone, we ranked each option in the categories described above and picked the drone that best fit the categories overall. The three factors that are most important to the project, which were the risk factor and hardware/software compatibility, are weighted double as if the drone is too risky to build or unable to satisfy the project requirements it is not a good option.

Drone	DJI Mini 3	Custom 10"	Custom 7"	Partial Custom 4"	Pre-Built 4"
Cost	4 (\$450)	1 (\$750)	2 (\$650)	3 (\$600)	3 (\$500)
Work to Fly	5 (Easy)	1 (Hard)	1 (Hard)	3 (Okay)	4 (Easy)
Software Customizability (x2)	2 (Bad)	5 (Great)	5 (Great)	5 (Great)	5 (Great)
Hardware Customizability (x2)	1 (Bad)	5 (Great)	5 (Great)	5 (Great)	3 (Okay)
Risk (x2)	3 (Okay)	1 (Risky)	1 (Risky)	3 (Okay)	5 (Little risk)
Fly Time	5 (51 mins)	4 (38 mins)	3 (30 mins)	3 (30 mins)	3 (28 mins)
Total	26	28	28	35	36

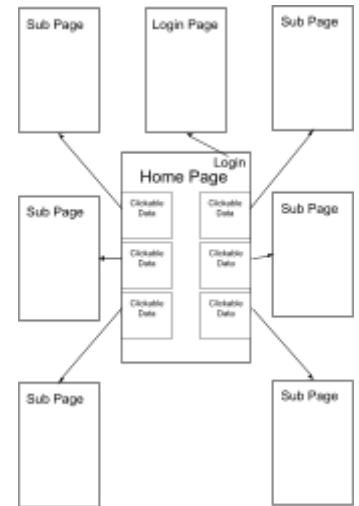
Ultimately, the two 4" drone builds were the strongest contenders: they give us great software control, since they run on open-source software, and both are generally not risky options relative to the fully custom drones or the DJI drone with limited software support. The only areas where they are not as good is in fly time, though most of the drone options we have were around the 30 minute mark, making them viable. The pre-built small drone (Flywoo Explorer LR, with parts listed in Section 3.7), was ultimately the least risky, safest option, while still including hardware and software that allowed the customization needed to operate the drone autonomously and for long distances.

4.3 Proposed Design

4.3.1 Overview

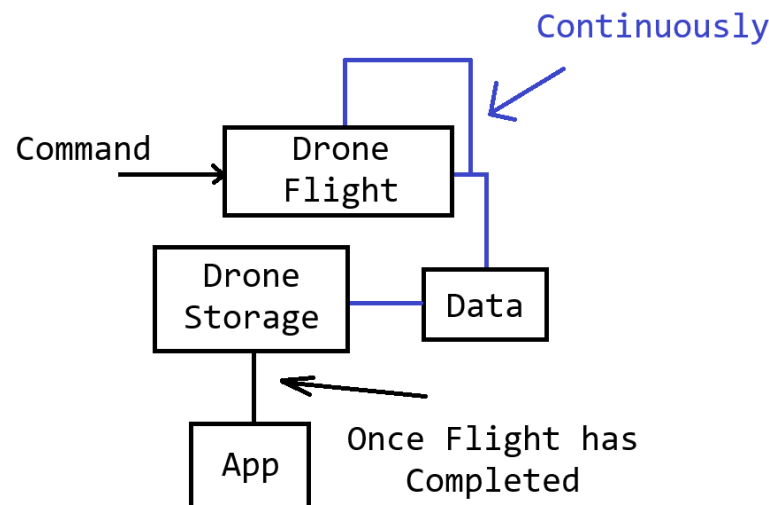
The drone system will take commands from a user, such as creating a flight path or flying a flight path, and will capture data by using its sensors, such as a camera or a GPS, to provide information to the user once the flight is completed.

The frontend design will consist of a main “home” page, as well as several sub-pages. The home page will display overviews of the most important collected data, which can be found on each of the sub-pages. By clicking on this data, the app will direct you to the respective sub-page for more in-depth information. The app will also feature user accounts, allowing each user to view their recorded data and identify any issues with their crops.



4.3.2 Detailed Design and Visual(s)

The design begins with the drone system which is configured with a GPS module, camera, and on-board Raspberry Pi Zero. The Raspberry Pi will communicate using our software with the open-source flight controller software, INAV, which flies the drone, to get telemetry data and set waypoints for the drone to fly to, using a serial connection and the MSPv2 protocol. While the drone is flying, telemetry data from INAV back to the Pi will note where the drone is, and at certain waypoints the Pi will take an image with its attached camera and store it to an SD card. The drone will continue to capture data until the battery runs low and it needs to return prematurely or all waypoints are reached. Once the user wishes to process the data, the user can pull the data (images and telemetry) from the SD card from the drone which will be read by the frontend.



For the frontend, we are using an Electron-Vite configuration using React. This allows us to work in TypeScript as well as JavaScript, and use CSS files for styling. Electron and React provide a preview of the page that updates in real-time and can be resized. On the home page, we plan to feature dynamically updated data images from the most important sections of each separate page on the website. Each image will direct you to its respective page when interacted with. Additionally, we will also include a navigation bar at the top of the screen. The website will also feature user accounts so that each user can only view their respective data, and we will have “log in to view your content” over data fields on the home page if the user is not logged in. User passwords will be censored as they are entered, and the website’s entry fields will adhere to cybersecurity guidelines for websites.

4.3.3 Functionality

In a real-world context, our design is intended to minimize the amount of work required by the user while providing an intuitive experience across operations. Upon opening the application, the user logs into their account and is presented with a dashboard displaying past flights, including telemetry data, flight paths, surveyed locations, and notifications about any detected field issues.

From the dashboard, the user can navigate to the flight planning screen to configure a new field survey. Here, the user defines the desired field they want to survey and sends a signal to the drone to initiate the survey.

After the flight is completed, the user uploads the collected data—such as telemetry logs and captured images—via a USB cable or microSD card connection to the device running the flight planning and overview software.

Once uploaded, the frontend application sends the data to the backend API, which:

1. Stores the data on the server.
2. Interacts with an AI-based image processing microservice.
3. Returns the processed results to the frontend.
4. Synchronizes the data with the cloud database for backup.

When processing is complete, the user can view the results within the application. The UI presents the analyzed field data, highlighting crop health, growth status, and any detected anomalies (e.g., signs of disease). This process aims to ensure users can easily plan, execute, and review drone surveys with minimal manual effort.

4.3.4 Areas of Concern and Development

Our current design satisfies the functional requirements outlined by our advisor/client. The system is divided into separate components that aim to automate the field surveying process. The user interface is designed to be intuitive, interactive, and easy to navigate, while the backend architecture is modular and supports parallel task execution to ensure efficient data processing.

The primary concern at this stage is the hardware configuration, whether the selected battery, given the project's limited budget, will provide sufficient capacity to complete a full field survey and capture all required data. Although our proposed hardware meets all theoretical functional and non-functional requirements, real-world performance can vary. Environmental conditions such as low temperatures may reduce battery efficiency, and strong winds may affect the drone's stability and the quality of collected data.

We have also identified minor constraints, including limited internet connectivity in rural field environments. To address this, we plan to implement an in-memory data storage mechanism that allows temporary offline operation and delayed synchronization with the cloud once connection is restored.

Our immediate development priority is to acquire the required hardware components so we can begin integration testing and refine data collection and transmission processes. Due to additional safety considerations that needed to be made and approved, as well as limited parts availability, this has taken longer than anticipated, but we now have the necessary parts and made sure we were not blocked while waiting for this, working to get as much done as possible the meantime so we can move quicker once the drone is assembled.

4.4 Technology Considerations

Depending on the drone system that we receive we are implementing a few different technologies. A couple of them are distinct by the size of the frame—spanning from 4-7” which allows for greater carry while weighing more, but will allow for higher battery or being able to sustain wind conditions more. The higher the frame the more expensive the parts become. They are all deployed on relatively similar drone parts from their frames since the power needed varies. However they all contain motors, props, flight stacks, a computer, and additional modules to fulfill the technical requirements of the design.

Given that we can either receive a custom/partially custom drone kit or a pre-built, the process in how they will fly varies. The ability to be able to program the drone to our specifications is more feasible under the custom drones as they do not restrict us to proprietary SDK’s while allowing us to use open-source software that will expedite the process for our application. However building a custom drone will involve slightly more risk as although we have extensively researched parts and drone systems, pre-built drones have higher reliability in working immediately and throughout the project.

Being able to process the data or images that we receive from the drone will also be unique to custom drones or pre-builts. For pre-builts, there seems to not be a distinct way to be able to freely process the images that the camera captures. Meanwhile, the custom drones will be able to utilize a computer and an SD card to extract images and process on our app.

4.5 Design Analysis

Getting drone hardware has been more difficult than anticipated, and was delayed significantly relative to our initial timeline. Our original drone builds were deemed too risky due to their highly custom nature, requiring us to start over and pick a drone that was mostly pre-built, and we additionally had to implement several safety constraints. The final build was also set back a few weeks due to shipping delays. In the meantime, we implemented as much software as possible referencing the documentation for the open-source software the drone’s flight controller runs and the Raspberry Pi the drone will use for data collection control, and we now have the parts necessary to assemble the drone and begin testing this work.

For the frontend, we have chosen an Electron-Vite configuration with React. This allows us to compile for Linux, Mac, and Windows. Our initial design plan was to use a .NET MAUI configuration; however, after discovering that .NET MAUI does not support compiling on Linux, we had to modify our design plans to maintain our CI/CD pipeline within the Git repository. This pipeline is crucial for testing purposes, so it was necessary to keep it intact to ensure that every feature and frontend/backend communication worked properly. Front-end features we have implemented so far include a homepage with highlights of the telemetry data that link to their respective pages, as well as login fields that serve as logins for user accounts and as a means to test our database connections. In the future, we plan to design and add pages for each of the separate telemetry data sets, as well as the database connections required for those pages.

5: Testing

5.1 Unit Testing

- For the frontend, we are using the React Testing Library for Unit testing. We are testing individual windows, interfacing between windows (such as changing views), and also testing for both regular and erroneous inputs. It sounds simple, and usually the tests are, but as the frontend continues to grow in scale, the number of different paths a user can take or buttons they can click rises significantly, requiring more unit tests.
- The drone software, written in C++, utilizes the Google Test library for automated unit testing. The major sections of the software being tested are the serial communication code (between the flight controller and the on-board computer), the camera controller code, the flight plan generation code, and the telemetry parsing and collection. Within each of these sections, individual functions are tested for common expected cases and possible edge cases, as well as for larger classes as a whole, to ensure that class state and methods interact correctly.
- The backend uses xUnit as its primary testing framework within the .NET environment. All tests are organized under the AutomationDrone.Tests project, which includes:
 - **Base/** – shared setup utilities for test classes
 - **Services/** – unit tests for core service-layer logic
 - **TestUtils/** – test data factories and helper objects

Service-layer tests focus on validating logic, data transformations, and interactions with the data layer. SQLite (in-memory) is used during testing to simulate database behavior and constraints, ensuring that service operations behave correctly without requiring a live database.

5.2 Interface Testing

The frontend UI is rendered using React for seamless testing and development. This allows frontend designers to make changes in the code and see their effect immediately, to determine if the output is what they envisioned or not.

The backend API is tested using Postman to simulate network POST, GET, PUT, and DELETE calls to the specified endpoints. This allows both the frontend and backend to determine which input fields are required for each database call, as well as verifying that their API calls will function correctly before they are implemented into the codebase.

The PostgreSQL database where we store the flight data is also viewable via the VM that we were provided by the university, and this allows us to view the data that we POST and PUT to make sure we are both sending the right data and also that our GET requests are receiving the correct data from the right endpoint.

5.3 Integration Testing

- Integration between the frontend and backend occurs through API requests, where the frontend sends flight and location data to be stored in the database. Testing for this involves simulating API requests using Postman and then replicating the same format as those successful requests to integrate into the frontend functionality. This is then tested at a later time individually to ensure functionality still behaves within the expected bounds, and if errors occur, that they are handled properly.
- Within the drone software, the interaction between all major components is tested end-to-end using our Google Test test suite with mocked interfaces. The software is used to communicate over serial with the drone flight controller and with camera hardware attached to the drone's on-board computer, so using these mocked interfaces allows us to test (to the extent possible) how the software as a whole works together with general inputs that we define, without needing to risk damaging the actual hardware with real-world tests.

The backend integration tests target the API layer, validating end-to-end request handling across controllers, services, and the database. Tests are implemented using xUnit for consistency with the existing .NET test environment.

These tests verify that the backend provides correct and stable responses to the frontend and the AI microservice. Critical integration paths include:

- **Flight Operations:** create, update, retrieve (single and list), and delete flights
- **Flight Data Operations:** create, update, retrieve (single and list), and delete flight data
- **Location Management:** create and delete locations

Each integration test issues HTTP requests against the API, ensuring the full request pipeline operates correctly, including routing, validation logic, service-layer interactions, and database access.

5.4 System Testing

For the AI vision subsystem, system testing will focus on validating the end-to-end pipeline from drone image input to the growth/weed/pest outputs consumed by the backend and dashboard. The goal is to ensure that, given a UAV image, the system produces correct detections and growth estimates, in the expected format, within the required latency.

At the unit level, we will test individual Python components, including image preprocessing (resizing, normalization, and tiling), model loading, inference wrappers, and post-processing logic (thresholding, class mapping, and growth percentage calculation). These tests will be implemented using pytest and will utilize small, fixed test images and mock model outputs to verify correctness and handle edge cases.

Interface tests will verify that the AI microservice API matches the contracts defined with the backend, including the correct HTTP method, request schema (image or image URL, crop type), and response schema (classes, bounding boxes/masks, confidence scores, and growth percentage). These tests will use automated API calls against a staging instance to ensure backward compatibility as the model evolves.

Integration/system tests will run the full pipeline using a held-out set of UAV images and ground-truth labels. For each test batch, we will compute accuracy/IOU and compare against target thresholds defined in the requirements. We will also measure end-to-end latency (from image received to JSON response) to verify real-time constraints. Model-level evaluation will use the built-in YOLO/Ultralytics validation tools, while higher-level system tests will be scripted in Python and integrated into our CI pipeline.

5.5 Regression Testing

All changes made to the project are posted to the Git repository as merge requests, which are then reviewed by other team members before being merged. The repository has CI pipelines set up, which detect the specific section of the project where changes were made (frontend, backend, drone software, or Terraform) and run all available automated tests on it, reporting the results back to the GitLab interface and blocking the request from being merged. Running the entire test suite for any change ensures that earlier features, which were previously unit/integration tested, will run again with the changes, allowing us to detect unintentional changes to earlier behavior.

Critical system features, as dictated by our project requirements, will be covered by these automated unit and integration tests wherever possible, thereby reducing the risk of testing new changes against the actual drone due to undetected regressions.

- Drone serial communication is extensively tested with unit tests and a mock serial port implementation in software to ensure the drone flight controller receives the commands it expects. Additionally, it will test entire mocked flights with simulated data transmission between the mock flight controller and our drone software to ensure that changes do not cause issues with the autonomous flight capabilities.
- The frontend's critical features, such as field waypoints and data transmission to the drone, will also include automated testing in a similar form, with unit tests on a low level and integration tests against the entire flow as much as possible.
- The backend is set up to automatically test that every API endpoint functions according to the project requirements and that data persistence and storage are correct.
- The cloud infrastructure configuration utilizes Terraform, and merge requests that modify infrastructure will generate a plan (detailing the changes that will be made if applied) in the CI logs. This plan is reviewed before being applied, which helps uncover unintentional changes to previously deployed infrastructure.

5.6 Acceptance Testing

To demonstrate that our design requirements are being met, our drone will accurately fly along a pre-designated flight path and take pictures of the crops in that field. These pictures will then be uploaded to our app's frontend, which will take the flight data and pass it to our backend database. From there, our database will send the crop images to the AI model, where the trained model will analyze these pictures to determine and returns correct growth stage and growth percentage. To involve our clients in the acceptance testing, we will reach out to farmers and ISU farms to gauge their willingness to test our product and provide feedback, whether based on their UI/UX experience or the accuracy of the AI model responses. This feedback provides an iterative approach to improving the current design.

5.7 Security Testing

When considering security testing for our project, we must ensure that we cover every aspect of our project. The components we have identified are the frontend dashboard, backend (API), drone software, AI microservice, and cloud infrastructure. Because the system can give the drone commands to operate autonomously, it is crucial to prevent unauthorized access to ensure the integrity of the flight instructions and the data/information itself. We can use STRIDE as a guide for a threat model for this project to identify the risks. From there, we can ensure that each threat is assigned a specific test case, ensuring our testing covers all relevant security requirements.

Backend security testing can be performed using Postman & xUnit, where we validate authentication, authorization, and input validation by sending malformed payloads, invalid tokens, and large numbers of requests to ensure the backend handles potentially malicious behavior correctly. The drone software can be tested through services like Google Test and mock serial interfaces, ensuring the onboard computer only accepts commands that are currently authenticated and formatted. For the frontend, React Testing Library and manual testing can be used to test secure token handling, form validation, and unauthorized page access. The AI microservice can have similar testing for payload validation and endpoint isolation checks.

For infrastructure, Terraform plans and CI-based security checks can verify that our cloud resources are configured correctly, adhering to the principle of least privilege. Static analysis tools can be used for this (such as SonarCloud, ESLint, or cppcheck), but probably won't be throughout this project.

Altogether, these efforts can ensure that security is a priority in our project and is implemented throughout the entire development process.

5.8 User Testing

Our project will undoubtedly involve extensive user testing, as it will be used exclusively by farmers. We want to ensure that our product flow, UI, drone, and everything else are usable, safe, and accessible to users without requiring advanced technical skills to understand how to use them. We will be focusing on asking farmers how they feel about the product, the UI, and the drone. Their input is invaluable to us as they are the target audience of our product. Once we have completed most of the product's functionality, we plan to solicit feedback from farmers, asking them if we can survey their fields and what they think about the product. Their suggestions will be taken seriously and implemented if possible. We want to create a product that will actually be used, and if our users aren't happy with it, then they won't use it.

5.9 Results

Thus far, we have successfully implemented a backend API and integrated it with the frontend user interface. We can successfully send and retrieve flight and location data, provided the proper flight ID is provided, allowing users to access only their own flight data. This will later be expanded to include authentication and user accounts, allowing us to track flight data per user rather than users per flight data.

Having received the drone hardware late into the semester, we were unable to spend much time testing with a flying drone. However the plans for testing a flying drone begin with us mounting the drone onto a test stand to test the pitch, roll, and yaw to make sure our PID loop has been controlled. Next is to test indoors while the drone is tethered to the ground to ensure that the drone does not accelerate into the air when the drone is outside. We will then test the drone indoors without the tether to test functionality before flying outside. There will be a kill switch implemented to the motors so that the drone can stop flying immediately in the event of unexpected behavior. Plans for a drone return after flying too far is considered as well. We have also implemented extensive automated/simulated testing of our drone software to the extent possible without the drone.

6 Implementation

Thus far, we have successfully implemented a basic frontend UI, complete with site navigation. These separate pages allow us to show a simplified breakdown of important information on the home page, and then link to the more in-depth pages when each breakdown is interacted with. We have also successfully implemented the Google Maps API for our users to mark out the size of their field that they would like the drone to patrol. This allows us to then plan the drone's flight path to maximize coverage of the subdivisions of the field, while also minimizing flight time (to conserve battery power on larger fields).

Home

ictures

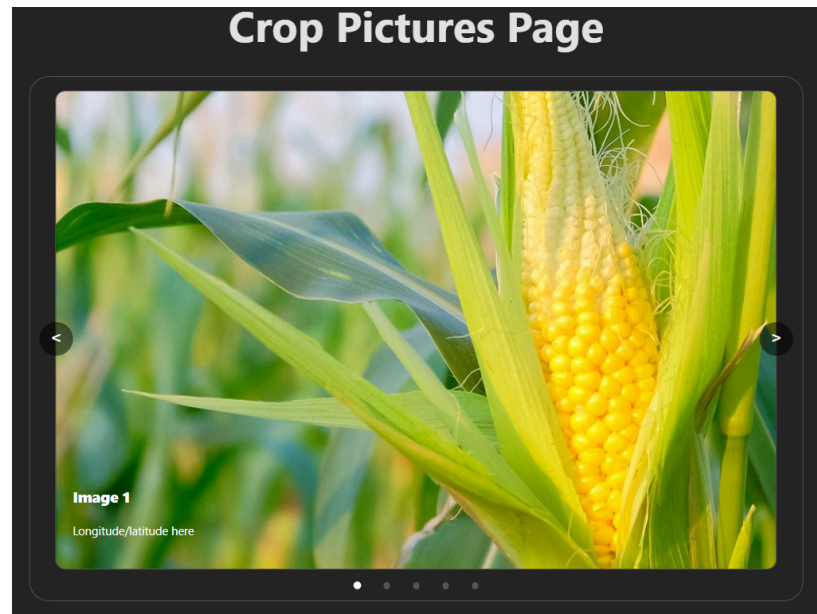
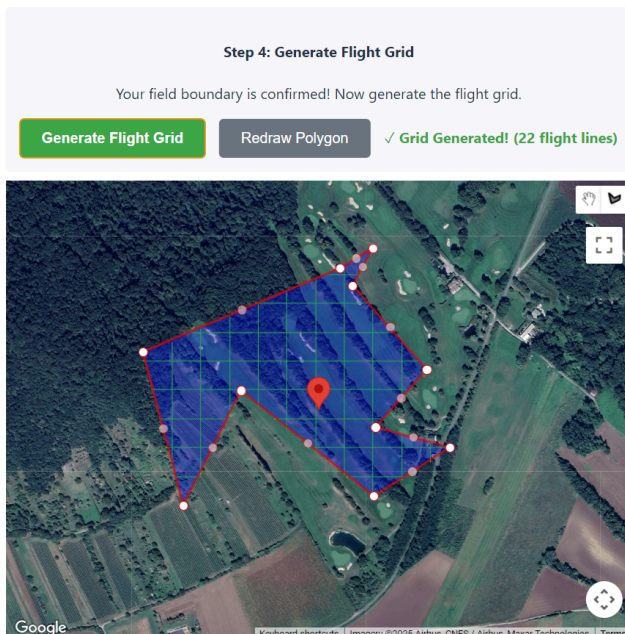
Enter your Address, City, State, then ZipCode

Format like this: 123 Main St, Springfield IL, 62701

Submit

Use Default Location

Crop Pictures Page



7 Ethics and Professional Responsibility

7.1 Areas of Professional Responsibility/Codes of Ethics

Area of Responsibility	Definition	Relevant Item from ACM CoE
Work Competence	The skills and effort to complete work at a high quality and with efficiency.	2.1 and 2.2, we're all trained in some ways to help contribute to different areas of the project. We've been making progress in all areas of our project while maintaining a high level of work.
Financial Responsibility	Take care of the parts and pieces that we spend money on, be frugal when purchasing.	1.5, by respecting the work required to make the parts for our drone, we are also respecting the money it costs to buy it.
Communication Honesty	Have open communication between each other and especially with Maruf about project progress.	1.3, we update Maruf every week with the progress on our project. If we're stuck or need help we tell him. We have a shared discord server where we constantly check in and update each other on progress, adding another layer to our communication honesty.
Health, Safety, Well-Being	Learn from other drone safety issues, take our drone build and testing serious while prioritizing safety over progress.	1.2, We've been in meetings and discussions with Dr. Jones about drone safety and what it looks like in our project. We've learned how to slowly iterate to test functions of our drone so that no one gets hurt.
Property Ownership	Don't damage buildings or crops with the drone.	1.2, avoiding harm is broad, but here we're applying it to avoiding harm of property. We haven't flown the drone but when we do we will implement safe guards and backups to ensure that crops and building aren't damaged by our drone.
Sustainability	Shut off the drone when done and don't have debris fall off the drone.	1.1, part of contributing to society also includes not leaving a trace. Our drone won't leave debris, pieces, or anything of that sort while flying over fields and taking pictures.
Social Responsibility	Only gather pictures of crops from farmers who consent to it.	1.6, while we haven't been able to take any pictures of crops yet due to it being winter, in the future we will respect the privacy and wishes of farmers.

An area that our team is performing well in is work competence. We're working in around five different areas of our project and all five areas are making significant high-quality progress. We've got specialists in every area to help ensure that we are producing competent work. One area that our team needs to improve in is communication honesty. This is not because there has been any dishonesty or communication, but rather there is the most room to improve on this area. We need to perform more thorough code reviews and also update each other more often on our progress. Our communication with Maruf is flawless so far.

7.2 Four Principles

	Beneficence	Non-Malificence	Respect for Autonomy	Justice
Public health, safety, and welfare	By improving crop health, can increase protection for crops.	Drone has safety features to prevent harm.	Drone will take commands at request of user. Apps will not take user data.	Healthy crops promote safe food production for multiple farms.
Global, cultural, and social	Food can be produced faster.	Follow flying laws.	Communities can modify drone.	Higher quality crops can be obtained for more farmers.
Environmental	The project helps the crops by identifying their potential diseases.	Reduction of chemicals that is used on crops, electricity for drone instead of heavy machinery.	Farmers can efficiently choose how they want to grow crops with the information obtained from drone.	Healthier crops will lead to healthier farms across communities.
Economic	Increase yield and reduce waste or crops that have been thrown out.	Reduce crop failure risk.	Drone is not proprietary, and user can modify as needed.	The jobs of people that may go out and perform this labor would be replaced by the drone.

The most important context-principle pair for this project is safety and non-malificence. In regards to an Autonomous Drone, there needs to be precautions taken to not cause any harm to the user. The blame would fall entirely on the team if there is an error that causes the drone to potentially severely injure a user or anyone nearby. We are working to ensure safety by implementing safety features such as a kill switch and rigorously testing flight capabilities of the drone to catch any unexpected flight patterns.

A broad context-principle pair in which the project could improve is global and non-malificence. While the team has taken steps to ensure that the process of flying a UAV is being met, there could be improvements to really secure the flight control so that nothing unexpected occurs with flight and local and national flying laws.

7.3 Virtues

Three virtues that are important to the team are cooperation, friendliness, and reliability. Cooperation is the sharing of knowledge and willingness to work with others. Cooperation has been supported in the team within the different portions of the project such as hardware, frontend, and backend. All these sub-teams work with each other to create a better version of the project than if we has not cooperated well. This extends to the entire team as there has been a willingness among the team to help out when needed. Friendliness is sharing a bond with others that is not just a camaraderie team type of relationship, but one that exceeds that where it is fun to work with the members of the group. This is always being highlighted whenever the team meets for the project as well as the “group bondings” the entire team participates in. The stronger bond that is shared between us, allows us to open up to each other more about any help or vulnerabilities that we have. The last virtue of reliability is the setting of expectations with the team which is then lived up to by each member. Reliability has appeared through the continuous attendance of team meetings by each member as well as the technical deadlines being met.

- Alex
 - Responsiveness
 - Obviously it is very frustrating when you are trying to work on a project and you need to get a hold of a team member, and they are not responding. Even often for quick questions it can put a hold on whatever you're doing. So being easily accessible and quick to react is very important to me.
 - Making sure I am always reachable and quick to action has been a priority for me this past semester. I always have my phone on me with notifications configured for my project group. I also make it my first priority to make any requested changes as soon as I am able. I also have made sure that there is documentation and READMEs that are current and can provide any additional information while I am away.
 - Initiative
 - Being able to be relied on to go above and beyond on the project and work ahead of where we are is something that can save a lot of headache further down the road. Taking that leap and doing more than what is required in the moment is something I value a lot.
 - Looking ahead in the calendar and seeing what goals we have set that I can complete now instead of waiting for some arbitrary deadline. Picking a goal and setting my own deadline to have it done to to then be able to present this progress to the team would be beneficial in my eyes.
- Caleb
 - Attentiveness
 - Attentiveness is important because without prompt response from each member of the team, code reviews and feature changes would take longer, delaying the project exponentially.

- I have demonstrated attentiveness by responding to requested frontend code reviews that pass our CI/CD pipeline within 24 hours of the request. I have also begun work on features that interact with the backend as soon as they are ready to proceed. In these ways, I help minimize downtime and help keep the team moving forward
 - One virtue that I still have to work on is moderation
 - Improving my moderation is important because I oftentimes struggle when the frontend UI isn't responding or reacting how I want it to. This can lead to me making the situation worse by trying a bunch of random ideas to see if any of them work, and then adding bloat to the project by forgetting what I've tried.
 - To improve my skills in moderating myself, I need to work on taking a step back when the project isn't adapting how I think it should be, taking a moment to analyze what I've already done, then proceeding with those new changes in mind.
- Srdan
 - Curiosity
 - Throughout this project I let curiosity drive my contributions from a technical and non-technical perspective for the project. Taking on a technology I have never used previously drove my passion for researching, implementing, and testing various systems and approaches to solving the problems our team set as a goal.
 - Over the course of this project, my curiosity drove me to consistently put out suggestions for the rest of the team, try to learn how different groups within our team were implementing their solution, and what improvements we can make in various parts of our system that would benefit us in the end.
 - Self-Control
 - Having self-control is important in order to maintain professionalism, but also help better guide the rest of the team. I had lacked self-control in certain cases when I got ahead of myself, and wanted to implement various functionalities for this project that were not realistic within the timeline we were working with.
 - In the future, I intend to do a better job hearing my teammates' thoughts on why something might not be the best fit for us or why we should take a different approach to implement something differently from what I had imagined, which will lead to a better overall development experience.
- Andres
 - Humility
 - This is important to me because I like to view myself as never being the smartest person in the room so I can continuously learn from others that are far greater than myself. With this perspective, it allows me to ask questions and multiple of them, while staying humble to my own work and knowledge to reciprocate any knowledge given towards me in a group project.

- I have received guidance from my peers for multiple topics of the project. Since I am not familiar with many aspects of the project, they have taught me how the project works in regards to their expertise. Even in my own area of hardware, others in this project have shown me a lot that inspires me and helps me out.
 - Dedication
 - Dedication is important to me because it shows that there is passion into a project and when there is passion, a great project can be made. While I was and still am interested in the project, other classes and a lack of discipline has not led me to working as much on the project as I could have this semester. This leads to me not showing that much work in what we could have shown at the end of this semester.
 - To demonstrate dedication I need to put time into the project and even if there is a hold in the project that is outside of our hands like what has gone on during the semester, I need to continue to surround myself with ideas of the project and give a helping hand to other sections of the project.
- Jongmin Lee
 - Perseverance
 - Why is it important to you?
Perseverance is essential because complex systems like, AI vision pipelines, rarely work on the first attempt. Progress often depends on steady iteration, debugging, and learning from failed attempts. Maintaining persistence ensures that even difficult technical challenges are eventually overcome.
 - How have you demonstrated it?
I have consistently demonstrated perseverance through the AI model development process. Despite dataset issues, training failures, and the need to restructure our model goals, I continued refining the dataset, reorganizing labels, and retraining models until performance stabilized. I also pushed through long training cycles and environment setup problems on the cluster to ensure the AI pipeline moved forward.
 - Proactive Communication
 - Why is it important to you?
Proactive communication is important because our project relies heavily on cross-team coordination, especially between the AI, backend, and frontend groups. When updates or blockers are communicated early, it prevents wasted work and makes integration much smoother. Improving this virtue ensures the team stays aligned and avoids bottlenecks.
 - What might you do to demonstrate that value?
I can demonstrate more proactive communication by sharing consistent progress updates in our team channels, especially when running long experiments, discovering issues with data, or identifying changes that may affect the backend or UI. I also plan to signal blockers earlier and request help instead of trying to solve every issue alone. This will improve team workflow and reduce integration delays.

- Camden
 - A virtue I have demonstrated is craftsmanship.
 - This is important to me because I always want to produce the best possible work I can, spending lots of time researching and making sure everything is well-thought-out, maintainable, and reliable – least likely to cause any issues down the road, and most likely to help the rest of the group in their own work.
 - I have worked hard this semester to produce well-documented systems that show an attention to detail in their design, making sure the team can understand them. I have also done lots of additional work to improve the development experience with thorough CI pipelines, well-automated and tested systems, and lots of documentation.
 - A virtue I want to be better at demonstrating is collaboration.
 - I have a tendency to work heads-down on a task I have through to completion, but in a large group it's important that we all spend time working through designs together and splitting up tasks to ensure that the work we individually complete will work well with the larger product and that everyone gets to have a part in it.
 - I want to be better at splitting up tasks and discussing their implementations with the group in our weekly meetings so that each part of the project can be refined and made better in the context of the whole project, and supporting team members working on other parts of the project.
- Z
 - A virtue I have demonstrated is willingness to fail.
 - This virtue is important to me because it shows that I want to learn and grow as a developer. One of the necessary traits to have as a software engineer is to want to learn. The chase for knowledge and skills is a major motivator for me as an engineer. I am happy to have demonstrated this virtue throughout this semester.
 - I haven't worked on a React frontend before so being assigned to frontend was a completely new experience and there has been a lot of failing, but also a lot of learning. I think the biggest way I've shown this virtue is the progress I've made on the frontend and the questions I've been asking of my teammates that are more familiar with React.
 - A virtue I want to be better at demonstrating is moderation.
 - Moderation is an important virtue because lack of moderation means that you are either putting in way too much effort or way too little. In my case it's a bit of both. Due to my schedule outside of school I have weeks where I contribute a lot to the project and weeks where I do a lot less. This leads to me being an inconsistent teammate which I don't want, which is why I chose this virtue to be better at.
 - I can show improvements in my moderation by being more consistent, setting time aside to work on the project at the beginning of the week. I can also track my improvement in moderation by keeping a record of my merge requests. I want to get one merge request in a week, getting something important done every week, but not doing too much, is how I can get better at moderation. I believe consistency is key in life and I want to strive to be better in that aspect.

8 Closing Material

8.1 Conclusion

For the hardware portion of the project, the drone parts have arrived but have not been assembled yet. Therefore, flashing INAV onto the flight controller and configuring the drone and controller has not been completed yet. Portions of the code have been written and tested while others have not. Raspberry Pi image code has been tested and will be continuously updated as the project progresses with needs that arrive in the future. The code for taking images has been written but not tested yet, while storing the images onto the SD card has not been written yet. The goal for the drone is to autonomously fly a field given user input while taking pictures in response to the user. These images will be stored into the SD card of the Raspberry Pi. Once the drone has completed the flight, or when it recognizes that it is low on battery, the drone will return home with images on the SD card.

For the frontend, we have successfully set up a basic user interface, complete with separate pages for each of the main sub-sections that we want to deliver. We have additionally set up a Google Maps implementation for the user to draw the outline of their own crop fields. We have also begun to integrate user authentication into each of the API requests, requiring users to sign in to their account in order to view any data. Our goals for the frontend include fully developing each of the individual pages, having working user accounts and authentication, and allowing the drone to upload the pictures that it captures to the backend database. In order to do this, we still need to develop the drone-frontend integration, allowing us to actually receive the captured images and pass them to the AI model for analysis. We also need to finalize the authentication in order to maintain user's data privacy, as well as cleaning up the UI for each of the sub-pages. Our main constraint this semester was time. If we wanted to complete more of these features, it would've required us to sacrifice time on other features. This would be something to consider if we only had this semester to develop this project, since we would've had to weigh which features would be important enough to prioritize, and which lower-priority features should be shelved in order to complete the higher-priority features in time.

Our backend system made great progress throughout this semester. We successfully completed the core user and flight logic, which includes allowing the users to create predefined flight areas, which can be reused for flight plans, as well as flight data collection logic, which we intend to use for providing users necessary data regarding their crops. For security, we utilized the AWS Cognito service, in order to have an authorization process, to block malicious attempts of retrieving a user's data. Additionally, we set up our database using the PostgreSQL engine, to ensure robustness and scalability, while keeping history of our migrations that have the table relationship logic throughout the lifetime of the development process. We aimed to have our full system working to the point of having a prototype version to showcase the full user experience, including having all the backend functionality completed. Overall, the core logic and functionality of our backend system is implemented, including the user authorization, data persistence, communication between the client and AI service, setting up the core backend system and structure to be modular and able to build on to in the future.

For the AI, we focused on establishing the foundation for a crop growth-stage detection model by organizing datasets, defining labeling guidelines, and preparing the training workflow that will later integrate with the backend and frontend systems. Our goal remain to first achieve accurate growth-stage classification before expanding into more advanced tasks such as pest, weed, disease detection. Progress this semester was limited by lack of drone-captured imagery and delays in hardware availability, which prevented us from collecting field-specific data for training and validation. Moving forward, obtaining real drone images, finalizing the growth-stage dataset, and building an automated evaluation pipeline will be the most effective path to achieving a reliable and fully integrated AI microservice in the next development phase.

8.2 References

[1] HashiCorp, “*Terraform Registry: AWS Provider Documentation.*” [Online]. Available: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs>. [Accessed: Dec. 7, 2025]

[2] Microsoft, “*Entity Framework Core,*” Microsoft Learn, Nov. 12, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/ef/core/>. [Accessed: Dec. 7, 2025]

9: Team

9.1 Team Members

1. Caleb Zea
2. Andres Lopez
3. Srdan Kopunovic
4. Z Harvey
5. Camden Senneff
6. Alex Pallan
7. Jongmin Lee

9.2 Required Skill Sets for Your Project

- Frontend:
 - Typescript usage
 - CSS usage
 - How to use and implement backend APIs
 - Data security implementation
 - Data security best practices

- Backend:
 - C# usage
 - API creation
 - API implementation
 - Database connection
 - AWS knowledge
 - Data security implementation
 - Data security best practices
 - Terraform knowledge
- Drone hardware:
 - Part compatibility
 - Safety procedures
 - Raspberry Pi programming knowledge
 - Raspberry Pi hardware knowledge
- AI:
 - Model building
 - Data Collecting and labeling
 - Model training
 - Backend integration

9.3 Skill Sets covered by the Team

Data security, AWS setup, Terraform: Alex

Backend API, User Authorization, AI backend integration: Srdan

Frontend API integration, Typescript and CSS, frontend-backend API integration: Caleb

AI model building, dataset training, dataset labeling, dataset organizing: Jongmin

Part building, safety procedures: Andres

AI backend integration, Raspberry PI code, Raspberry Pi hardware: Camden

Typescript, CSS, frontend-backend API integration: Z

9.4 Project Management Style Adopted by the team

We are using an Agile management style

9.5 Initial Project Management Roles

Camden: Technical planning and organization

Andres: Hardware

Alex: Cyber Security (encryption, auth, etc)

Caleb: Frontend

Jongmin: AI, Frontend

Z: Software jack of all trades

Srdan: Backend, Middleware

9.6 Team Contract

Team Members:

- | | |
|--------------------|-------------------|
| 1) Caleb Zea | 2) Camden Senneff |
| 3) Srdan Kopunovic | 4) Z Harvey |
| 5) Jongmin Lee | 6) Alex Pallan |
| 7) Andres Lopez | |

Team Procedures

- Day, time, and location (face-to-face or virtual) for regular team meetings:
 - Advisor meeting on Wednesdays at noon
 - Full team meetings 11am Saturdays, in-person preferred (location TLA), online as needed
- Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
 - Discord
- Decision-making policy (e.g., consensus, majority vote):
 - Majority vote

- Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
 - Take turns, alphabetical order (Andres > Caleb > Camden > Min > Srdan > Z > Alex)

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

- Be there and be on time, unless communicated with enough advance notice, ideally notifying the group at least 2 hours prior to meeting.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

- Commit to the team assignments that are requested each week and successfully accomplish them. However, if they cannot be completed, report why, and be prepared to have teammates help you resolve any issues.

3. Expected level of communication with other team members:

- Asking for help from the team when/before it is needed. Team members should communicate how they are planning to complete their team assignments with their role members and when they have completed their assignments to the team.
- Communicate if not able to attend meeting/lecture (at least 2 hours prior).

4. Expected level of commitment to team decisions and tasks:

- Actively participate in meetings and decision making.
- Ensure all tasks are completed by the set due dates, and if to communicate any issues that could potentially slow team progress

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

Camden: Technical planning and organization

Andres: Hardware

Alex: Cyber Security (encryption, auth, etc)

Caleb: Frontend

Min: AI

Z: Software jack of all trades

2. Strategies for supporting and guiding the work of all team members:

- Hold each other accountable
- Keep track of each other's contributions and reach out if someone is falling behind
- In the weekly meetings, talk about what we want to get done and what we need from each other to accomplish that

3. Strategies for recognizing the contributions of all team members:

- Have team members discuss their contributions of the week and successes of or why they have not accomplished their task
- Check off sections of the planned timeline for the project, including each contributor's accomplishments.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

- Andres - Embedded Systems, made drones fly before, taking machine learning. I like to code in C. I like hardware as well as background with circuits.
- Srdan - Embedded Systems, solid amount of backend experience, skilled in Python for processing large data sets, good hardware knowledge and an ability to properly set up separate systems to communicate with one another.
- Caleb - Embedded Systems, backend database knowledge, frontend functions, some security knowledge.
- Min - AI, specifically language model and machine learning, Python for AI and data, frontend experiences, preferred language is Java, and Python.
- Camden - I have a lot of experience developing secure backend systems and APIs, and I am additionally comfortable writing software for embedded systems, developing frontend applications, and working with AI. I am most experienced in Java, Python, C/C++, and Go.
- Alex - Python for data, Cyber Security concepts and applications, Infrastructure management (Cloud, physical systems infrastructure), Frontend Development, FAA Certified Remote Pilot.

- Z - Embedded Systems, frontend and backend experience in Java, Angular, C++, Verilog. No matter what stage of development we're in I'll be able to help further our progress. Confident in everything except pure hardware.

2. Strategies for encouraging and support contributions and ideas from all team members:

- Create a concrete plan of execution
- Support each team member when they need help
- Collaborate for proper implementation
- Communicate well with members working on connected parts of the project

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

- Bring it up during a team meeting, and the group as a whole can discuss how we can fix the problem
- If there is a larger-scale issue, then report to Dr. Ahamed

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

- Having a prototype that has some if not all of the features implemented to some capacity.
- Collaborating together to create a design that we can be proud of

2. Strategies for planning and assigning individual and team work:

- Define frontend/backend roles to separate into smaller teams for better coordination.
- Breakdown tasks within those teams and assign them to individuals.
- Assign larger tasks as a whole team

3. Strategies for keeping on task:

- Creating a rough timeline of each task, and goal.
- Working on the project in phases, with the goal of implementing smaller sections of the program with each other, before moving on to other sections.
- Proper communication, use of version control software, and anything else that can support us in completing our work.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

- Any repeated failure to adhere to the contract will be discussed in our weekly team meeting with the group to attempt to come to a resolution.

2. What will your team do if the infractions continue?

- Discuss repeated infractions with Dr. Ahamed

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) Caleb Zea	DATE 12/5/2025
2) Camden Senneff	DATE 12/5/2025
3) Jongmin Lee	DATE 12/5/2025
4) Srdan Kopunovic	DATE 12/5/2025
5) Z Harvey	DATE 12/5/2025
6) Alex Pallan	DATE 12/5/2025
7) Andres Lopez	DATE 12/5/2025