

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
 - 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	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

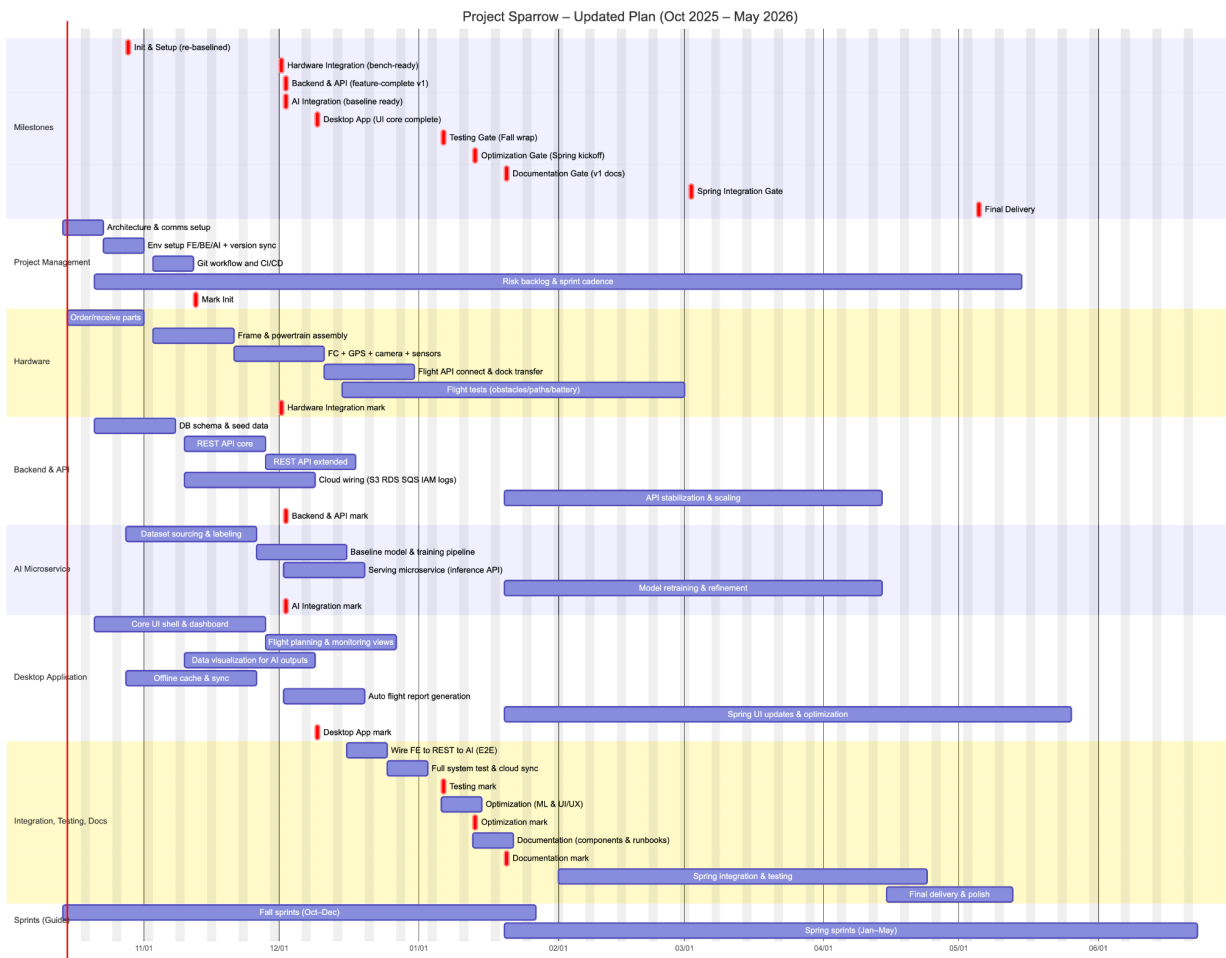
- Drone achieves autonomous navigation across a predefined area with $> 90\%$ completion success.
- AI model meets target accuracy ($\geq 70\%$) requirements.
- Backend/API passes testing and satisfies all data requests

- 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 (Alex)



3.5 Risks and Risk Management/Mitigation(Jongmin Lee)

Our project involves the integration of multiple complex systems — drone hardware, onboard AI vision models, cloud connectivity, and a farmer dashboard. Because of this, several technical and operational risks have been identified across our major tasks. Each risk is evaluated by probability and severity, with mitigation strategies proposed for those exceeding 0.5 probability or high potential impact.

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.

Agile Risk Tracking

Risks will be re-evaluated during each sprint.

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

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be the projected effort in total number of person-hours required to perform the task.

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	40

3.7 Other Resource Requirements

Identify the other resources aside from financial (such as parts and materials) that will be required to complete the project.

Hardware Requirements

The drone used in this project is custom-built using components readily available online, with an approximate total cost of \$750. Our suggested drone build is as follows:

- 10"/425mm carbon fiber quadcopter frame (\$30)
- 4x 3115 900kV motors (\$96)
- 4x 10"x5" props (\$21)
- 65A peak F7 Pro Betaflight flight controller (\$143)
- 8000mAh LiPo battery (\$153)
- Raspberry Pi Zero 2 W (\$20)
- Raspberry Pi HQ IR-cut camera (\$67)
- NEO-6M GPS board (\$7)
- ELRS receiver antenna (\$21)

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

- 6S LiPo battery charger (\$35)
- ELRS RC controller (\$73) and 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