

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, 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 assess 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. 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. Real-time communication with the backend allows users to view live 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.

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

List key design decisions (at least three) that you have made or will make. These can include, but are not limited to, materials, subsystems, physical components, sensors/chips/devices, physical layout, features, etc. Describe how you made/will make these decisions and how they have affected or are likely to affect project success.

A major design decision we had to make for this project is which 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, deciding instead to compile with Electron.

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 time will be a bit worse than the custom build due to additional weight included from the factory. 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)	4 (Little risk)	5 (Little risk)
Fly Time	5 (51 mins)	4 (38 mins)	3 (30 mins)	3 (30 mins)	3 (28 mins)
Total	26	28	28	37	36

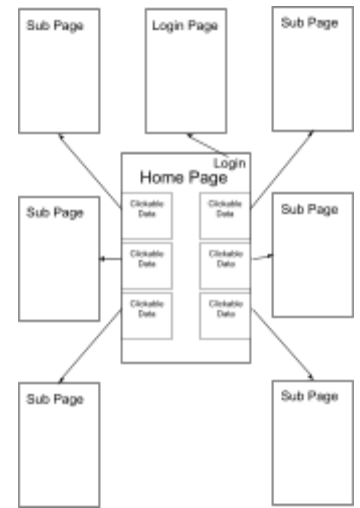
Ultimately, the two 4" drone builds were the strongest contenders: they give us great software control, since they run on open-source software, and both are generally not risky options relative to the fully custom drones or the DJI drone with limited software support. The only areas where they are not as good is in fly time, though most of the drone options we have were around the 30 minute mark, making them viable. The partially custom 4" drone won over the pre-built as it gives us better hardware customizability for our on-board computer and sensors and has a slightly better fly time with the weight optimizations.

4.3 Proposed Design

4.3.1 Overview

Provide a high-level description of your current design. This description should be understandable to non-engineers (i.e., the general public). Describe key components or sub-systems and how they contribute to the overall design. You may wish to include a basic block diagram, infographic, or other visual to help communicate the overall design.

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.



Frontend Diagram

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)

Provide a detailed, technical description of your design, aided by visualizations. This description should be understandable to peer engineers. In other words, it should be clearly written and sufficiently detail such that another senior design team can look through it and implement it.

The description should include a high-level overview written for peer engineers. This should list all sub-systems or components, their role in the whole system, and how they will be integrated or interconnected. A visual should accompany this description. Typically, a detailed block diagram will suffice, but other visual forms can be acceptable.

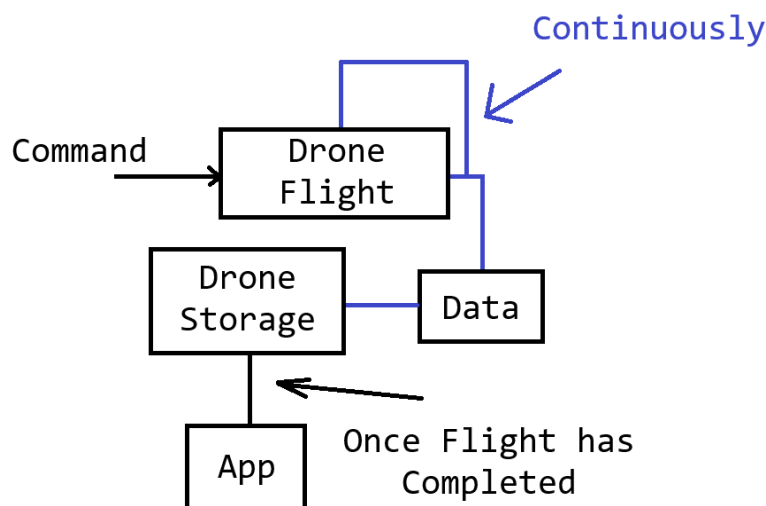
The description should also include more specific descriptions of sub-systems and components (e.g., their internal operations). Once again, a good rule of thumb is: could another engineer with similar expertise build the component/sub-system based on your description? Use visualizations to support your descriptions. Different visual types may be relevant to different types of projects, components, or subsystems. You may include, but are not limited to: block diagrams, circuit diagrams, sketches/pictures of physical components and their operation, wireframes, etc.

The design begins with the drone system which is configured with a GPS module, camera, and ELRS receiver. These components will work with the open-source software ArduPilot which allows a user to give a flight path on a map. The drone will receive telemetry data from the ELRS transmitter that the user is equipped with to fly the drone. Once the user requests to obtain information from the drone system, the video stream from the camera will be written to an SDCard on the Raspberry Pi. The drone will continue to capture data until the user requests to stop, while also flying a given flight path or flown manually. Once the user wishes to process the data, the user can pull the data from the SD card from the drone which will be taken to the app.

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 flight path 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. We also plan to consult our advisor/client to confirm the hardware delivery timeline.

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

Due to circumstances outside of our project, we have not been able to work with a drone yet. This has lagged a bit of the development of the physical side of the project. We have created firmware that will build onto the drone when we receive one however we have not been able to test this.

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.