



The AI-IADSSCHM

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

Overview

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

- Drone autonomously collects images from above a field
- Images processed by AI model
- Alerts, metrics, and flight planning with a desktop app

Requirements & Context

- Drone
 - Fly 30-45 minutes, can reconnect with charger if needed, sync data
 - Weather resistance
- Data
 - Our advantage versus existing products is drone is able to get more real time data vs satellite imagery
 - Can get daily updates on data to see more precise growth and pest information
- Farmers
 - Sole user group, important to build around them
 - Bright UI to be seen in sunlight, intrinsic UI for non-tech savvy farmers
 - Will ask farmers their opinions and feedback

Design Process

Planning

- Researching several options in detail
 - Drone safety: custom build, DJI, almost ready to fly
 - Custom: Not sure if parts will be compatible, safety concerns
 - DJI: already built and configured, minimal safety concern but not programmable
 - Flywoo: drone is built, can add Pi for code, less safety concerns

Experimentation

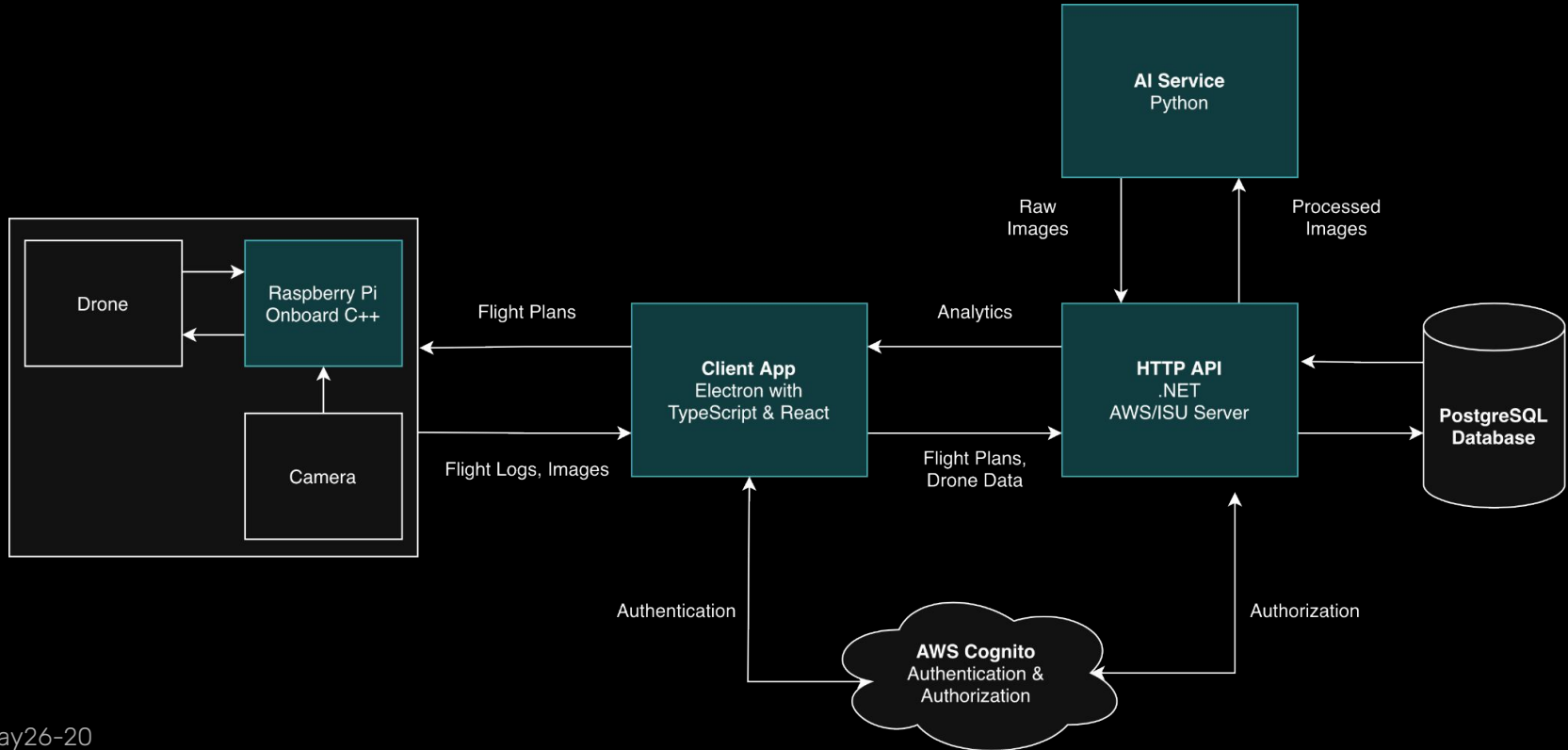
- Trying our ideas in practice
 - Frontend: MAUI to Electron, making own map to Google Maps JavaScript API

Proof of Concepts

- Our current stage: making functional prototypes

Design & Prototyping

System Overview



Drone

Flywoo pre-built drone running INAV

- Open-source drone flight control software
- Safe, easy to configure, more reliable than from-scratch, extensible
- INAV drones are easy to find, not expensive

On-board Raspberry Pi with C++ pilot software

- Raspberry Pi OS images created in CI
- Loads waypoints to INAV, collects telemetry, takes images, reads/writes data to SD card to be read by the desktop app
- Communicates with the drone over UART (MSPv2 protocol)

Drone

Accomplishments

- Raspberry Pi images with our software pre-installed
- Drone-to-Pi serial communication protocol
- Basic camera integration
- Extensive unit testing

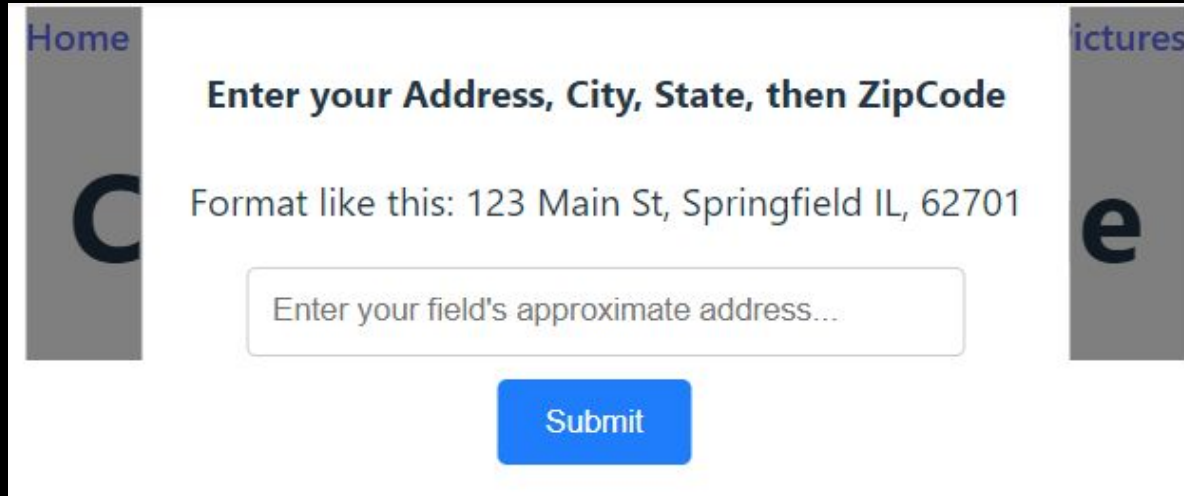
To-Do

- Frontend data transmission
- Waypoint transmission and telemetry storage
- Real-world testing

Frontend

Design	Development	Testing
• User login page redirects to the user's flight dashboard • Flight dashboard contains: crop health, flight path, telemetry data, etc • Want to be easy to understand and utilize	• Basic UI pages with simplified data, inserting placeholder images or data when necessary • Added a scrolling image carousel for crop pictures • Basic authentication in-progress • Implemented Google Maps API features for field maps and drone flight paths	• Manual UI testing using built-in React launcher • CI/CD pipeline deployment testing using the Gitlab CI/CD pipeline

Frontend (Pictures)



The image shows a web form with a white background. On the left and right sides, there are vertical grey bars. The left bar has the word 'Home' in blue at the top and a large black letter 'C' below it. The right bar has the word 'ictures' in blue at the top and a large black letter 'e' below it. In the center of the form, the text 'Enter your Address, City, State, then ZipCode' is displayed in a bold, dark blue font. Below this, a line of text says 'Format like this: 123 Main St, Springfield IL, 62701'. Underneath is a white text input field with a light grey border and the placeholder text 'Enter your field's approximate address...'. At the bottom center is a blue rectangular button with the word 'Submit' in white text.

Home

ictures

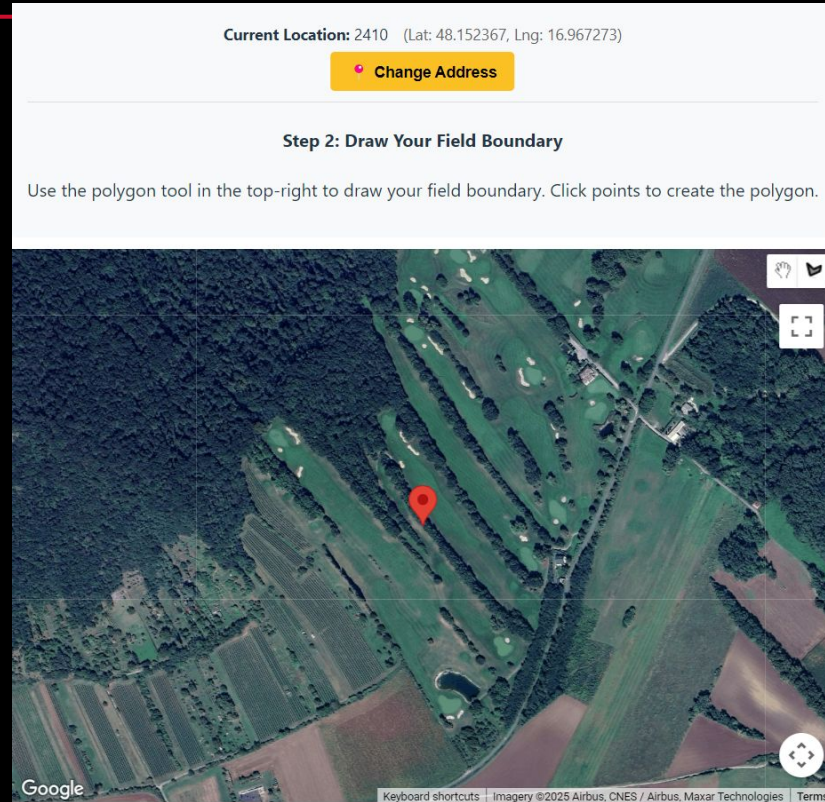
Enter your Address, City, State, then ZipCode

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

Enter your field's approximate address...

Submit

Frontend (Pictures)



Frontend (Pictures)

Step 3: Confirm Your Field Boundary

Are you satisfied with the polygon you just drew?

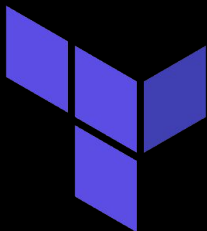
The polygon defines your field boundary. The flight grid will be generated within this area.

✓ Yes, Keep This Polygon

✗ No, Redraw Polygon



Cloud Deployment



- Infrastructure as code tool
- Lets you deploy, change, and destroy infrastructure



- Stores the state of infrastructure
- Runs upon push to main
- Plans and deploys the changes



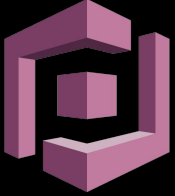
- Where the infrastructure is deployed
- Cognito (Auth), Database, Backend container (future)

```
main.tf 182 B
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5     }
6   }
7   backend "http" {}
8 }
9
10 # Configure the AWS Provider
11 provider "aws" {
12   region = "us-east-2"
13 }
```

```
database.tf 458 B
1 /*
2 resource "aws_db_instance" "awsdatabase" {
3   identifier      = "awsdatabase"
4   db_name         = "awsdatabase"
5   allocated_storage = 20 // max free storage
6   storage_type    = "gp2" // free tier
7   instance_class  = "db.t3.micro" // free tier
8   engine          = "postgres"
9   username        = "awsdatabaseusername"
10  password        = "secretpassword"
11  publicly_accessible = false
12  skip_final_snapshot = true
13 }
14 */
```

```
cognito.tf 957 B
1 // User Pool used to store users (like a database)
2 // Documentation: registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/cognito_user_pool
3 resource "aws_cognito_user_pool" "users" {
4   name = "users"
5 }
6
7 // App client that our software can use to interface with the user pool
8 // Documentation: registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/cognito_user_pool_client
9 resource "aws_cognito_user_pool_client" "client" {
10  name = "app_client"
11  user_pool_id = aws_cognito_user_pool.users.id
12
13  explicit_auth_flows = [
14    "ALLOW_USER_PASSWORD_AUTH",
15    "ALLOW_REFRESH_TOKEN_AUTH",
16    "ALLOW_USER_SRP_AUTH"
17  ]
18 }
19
20 resource "aws_cognito_identity_pool" "identity_pool" {
21  identity_pool_name = "Identities"
22  allow_unauthenticated_identities = false
23
24  cognito_identity_providers {
25    provider_name = aws_cognito_user_pool.users.endpoint
26    client_id     = aws_cognito_user_pool_client.client_id
27  }
28 }
```

Cloud Authentication



Amazon Cognito is the primary identity provider/manager.

User pool stores user authentication (username/password).

Opportunity to add multi factor authentication methods in the future.

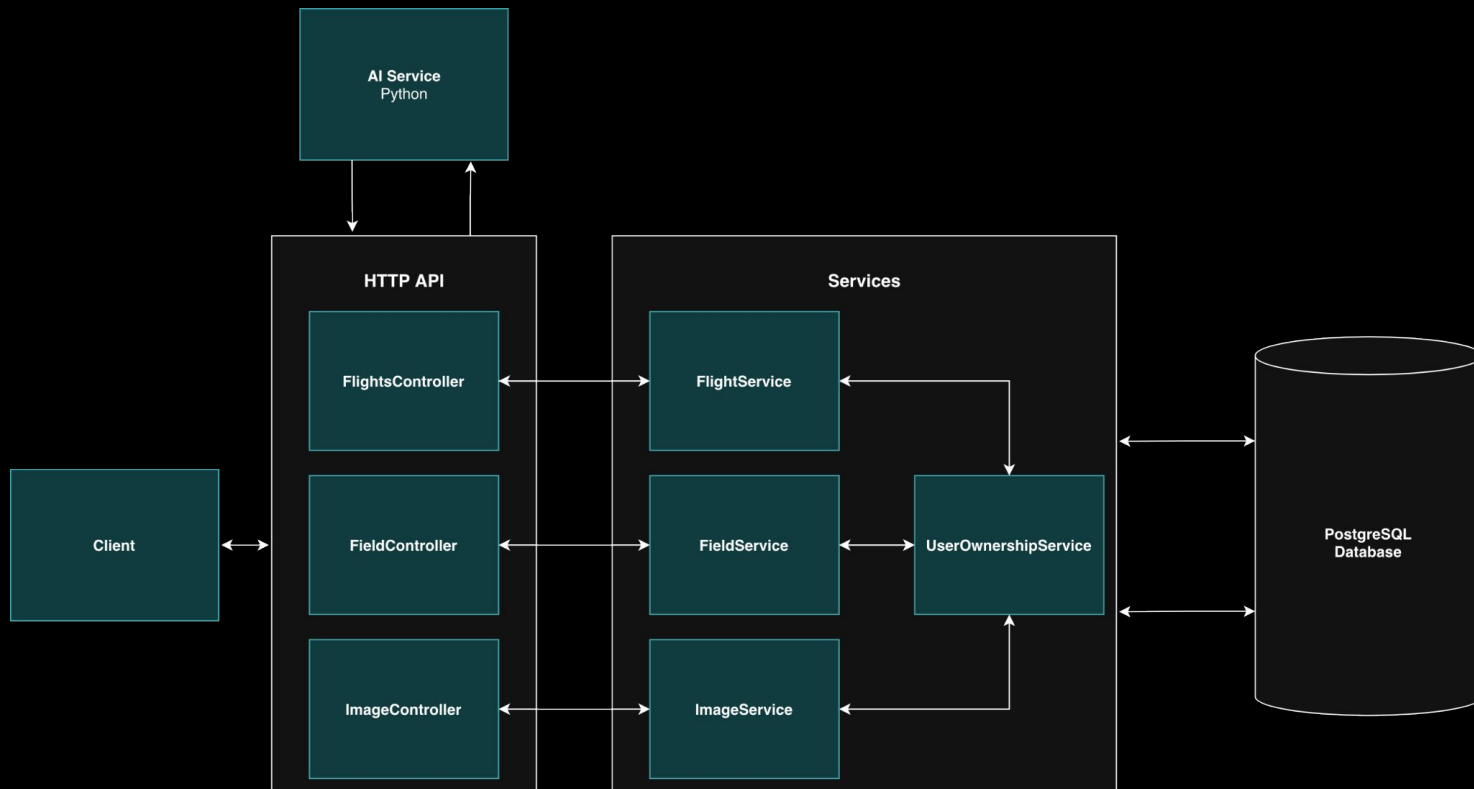
Identity pool stores user permissions:

- Manages user access to AWS services (storage, etc)
- Keeps user data separate

Frontend calls Cognito for *authentication*.

Backend calls Cognito for *authorization*.

Backend



Backend

Prototyping & Development:

- API: ASP.NET controller implementation for HTTP requests
- Service: Implementation for logic, data flow and error handling
- Data: Database configuration, table structures, and migrations
- Deployment: Development build on ISU Server, Production on AWS

Testing:

- Unit and Regression tests for proper logic functionality
- Integration Tests for correct interaction between modules
- End-To-End Tests for full system coverage

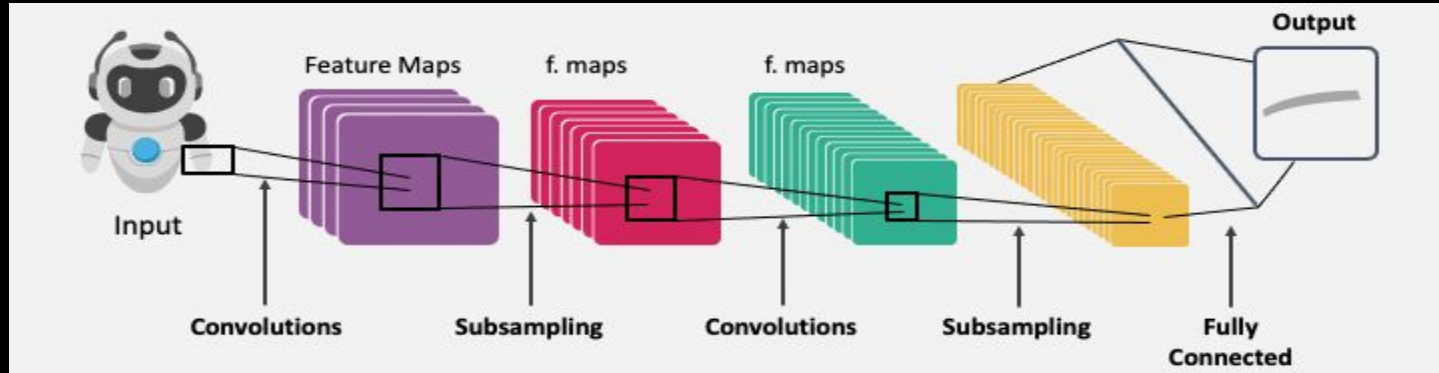
Model Architecture & Trade-offs

Model	Pros & Cons	Ideal Use Case
Standard CNN	+ High classification accuracy - No detection capabilities; cannot locate objects without expensive sliding windows	Simple Image Tagger: Sorting a photo album into "Beach" vs. "Mountain" without needing to find specific items.
R-CNN Family (Faster R-CNN)	+ High precision & tight bounding boxes - High Latency: Slow inference (seconds) creates a "laggy" user experience.	Medical Radiology: Finding tumors in X-rays where waiting 5 seconds for a precise result is acceptable.
Transformer (ViT)	+ Strong global context awareness - Resource Heavy: Requires massive data & compute for simple tasks.	Satellite Mapping: Processing high-res aerial imagery in huge offline server batches.
YOLO (selected)	+ Instant Inference: Processes images in milliseconds. - Minor precision trade-off for speed.	Instant Photo Analysis: My project, requiring immediate feedback when a user uploads an image.

AI - Vision

I. System Goal & Core Technology

- **Goal:** AI vision model to accurately **detect and classify the specific stage** of corn crops (and other target crops) across the field.
- **Core Technology:**
 1. **Model Type: Convolutional Neural Network (CNNs)** trained for multi-class classification and feature extraction.
 2. **Key Tasks:** The model analyzes images to identify and count visible features to determine the exact **Vegetative(V)** and **Reproductive(R)** stage.



AI - Data



The Need

An extensive, meticulously labeled dataset is essential. Each image is classified by its precise growth stage to train the model for high accuracy.



Data Input

Aerial imagery captured by the drone's camera, providing high-resolution visuals for stand counting and uniformity analysis.

The Standardized V-R Scale

Crop Growth



Vegetative (V) Stage
(e.g., VE, V3, V5)

Model detects leaf collars and plant size to quantify growth rates and track stand loss.

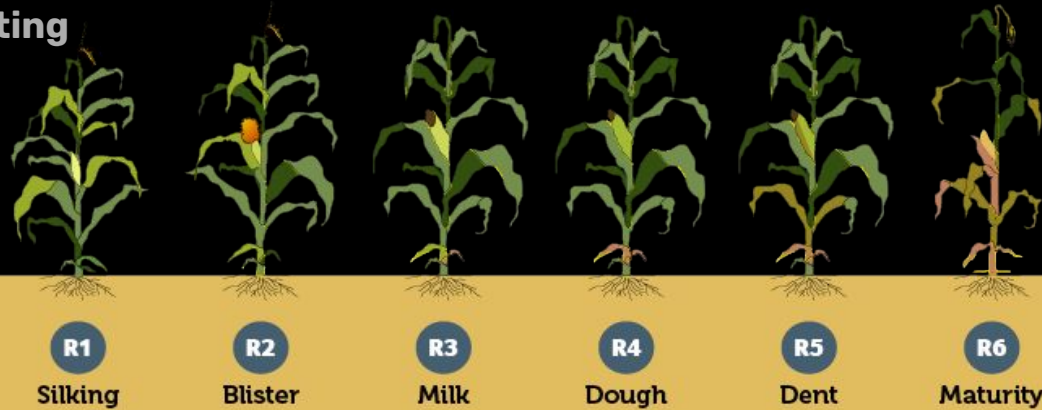


Reproductive (R) Stage
(e.g., R1, R3)

Crucial for maturity tracking and accurate yield forecasting.

AI - Data

- **Need:** An extensive, meticulously labeled dataset is essential, where each image is classified by its precise growth stage to train the model for high accuracy.
- **Data Input:** Aerial imagery captured by the drone's camera.
- **The V - R Scale:** Data is labeled using the standardized crop growth scale
 - **Vegetative (V) Stage:** (e.g., VE, V3, V5). Model detects leaf collars and plant size to quantify growth and track stand loss.
 - **Reproductive (R) Stage:** (e.g., R1, R3). Crucial for maturity tracking, and **accurate yield forecasting**



Conclusions

- Summary
- Next steps
 - Image passthrough
 - Finalizing authentication
 - Crop dataset for AI model
- Lessons
 - Research your tech stack and compatibility with other sections of your project before implementing (see .NET MAUI and CI/CD pipeline)
 - Drone safety
 - Getting data for crops in winter

Thank you

Questions
