agrivision

AI-Powered Smart Agricultural Control and Monitoring Platform

Supervised by: Dr. Mohamed Mead

Presented to the Faculty of Computer and Informatics
Department of Computer Science

Topics

1. introduction
2. Vision
3. Design
4. Implementation
5. Team & Workflow
6. Beyond the Build

Importance of Agriculture

Agriculture plays a vital role in our lives It is :

1. provides food:

- Agriculture is the primary source of food for both humans and animals.
- Without agriculture, there is no food security or social stability.

2. Providing raw materials for industry:

- Many industries rely on agricultural products, such as cotton (for textiles), oils, medicines, paper, and biofuels.

3. supports economies:

- Agriculture is a major source of income in many countries, especially developing ones.

4. creates jobs:

- It provides jobs in farming, food processing, transportation, and trade.

Challenges Facing Traditional Agriculture:

1.Resource Waste:

- Inefficient water usage.
- Excessive use of fertilizers, and pesticides without proper control.

2. Poor Problem Detection:

- Inability to detect pests and diseases early.
- Lack of effective monitoring systems.

3. Climate Change and Weather Uncertainty:

- Sudden weather changes negatively affect crops.
- Traditional methods cannot easily adapt.

4. High Costs with Low Return:

- Inefficient practices lead to more expenses and lower profits.

Challenges Facing Traditional Agriculture:

Key Statistics:

- Over 70% of global freshwater is used for agriculture — up to 60% is wasted due to inefficient irrigation. (UN Water)
- Crop diseases cause an estimated 20–40% loss in global food production annually. (FAO)
- Less than 25% of farms globally use any form of digital monitoring or automation. (McKinsey & Company)
- Climate variability has increased crop yield unpredictability by over 30% in the last two decades. (IPCC)

Topics

1. introduction
2. Vision
3. Design
4. Implementation
5. Team & Workflow
6. Beyond the Build

System Features

A unified, AI Powered platform for:

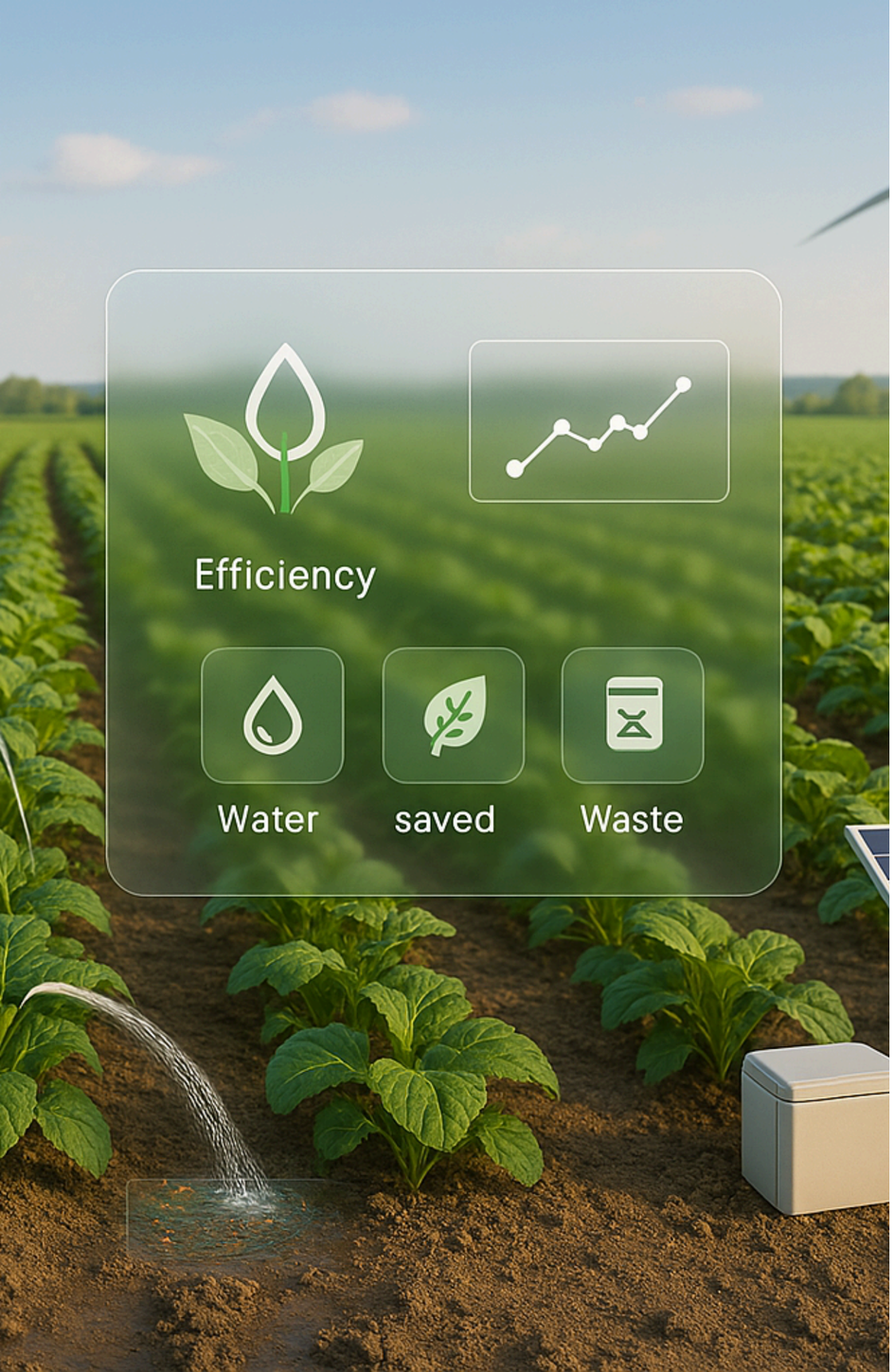
1. Farm Monitoring
2. Smart Irrigation
3. Crop Disease Detection
4. Collaborative task management
5. Reports & Analytics
6. Farm Resource Management

Research & Planning

- **Targeted at modern farms seeking smart, data driven management.**
- **Identified key gaps:**
  - Lack of real-time disease detection.
  - Limited multi-role access control.
  - Poor integration between AI and field data
- **Identified key gaps:**
  - Focused on irrigation, disease detection, and task collaboration.
- **Mapped key features to real needs**
  - Matched each problem to actionable solutions – e.g., smart irrigation to water waste, ConvNeXt and YOLO for disease detection.
- **Adopted clean architecture early and selected the technology stack**
  - To ensure clear separation of concerns and long-term scalability.

Platform Capabilities

- **Agrivision offers:**
  - IoT sensor integration.
  - AI-based disease detection.
  - ESP32-powered irrigation control.
  - Role-based access, task management and tracking.
  - Analytics & yield predictions.
  - Chatting system.
- **Web App:**
  - Advanced analytics and reports.
  - Centralized management on a large screen.
- **Mobile App:**
  - On-the-go access for users.
  - Enables real-time notifications.



Advancing Agricultural Sustainability

Agrivision aligns with multiple UN Sustainable Development Goals:

Goal 2: Zero Hunger:

- Enhances crop health and yield through real-time disease detection and smart irrigation.
- Reduces losses due to unmonitored failure and inefficiencies.

Goal 12: Responsible Consumption and Production:

- Data-driven farming reduces waste of resources like water, fertlizer, and energy.
- Helps farmers make informed, sustainable decisions.

Goal 13: Climate Action:

- Enables adaptive farming in farming response to environmental data and climate variability.
- Reduces unnecessary energy use with smart plug and sensor orchestration.

Agrivision Pricing

Basic

Free

- ✓ 1 Farm
- ✓ Up To 3 Fields

Features:

- ✓ Access To The Dashboard For Farm And Field Management.
- ✓ Basic Soil Health And Weather Insights.
- ✓ AI-Powered Disease Detection For Limited Usage

Get Started

Advanced

499.99 L.E / month

- ✓ Up To 3 Farms
- ✓ 5 Fields Per Farm

Features:

- ✓ All Free Plan Features.
- ✓ Advanced Analytics And Predictive Insights.
- ✓ Unlimited AI-Powered Disease Detection.
- ✓ Customizable Automation Rules For Irrigation And Sensor Integration.

Get Started

Enterprise

Custom

- ✓ Unlimited Farms
- ✓ Unlimited Fields Per Farm

Features:

- ✓ All Advanced Plan Features.
- ✓ Dedicated Account Manager For Personalized Support.

Contact Sales

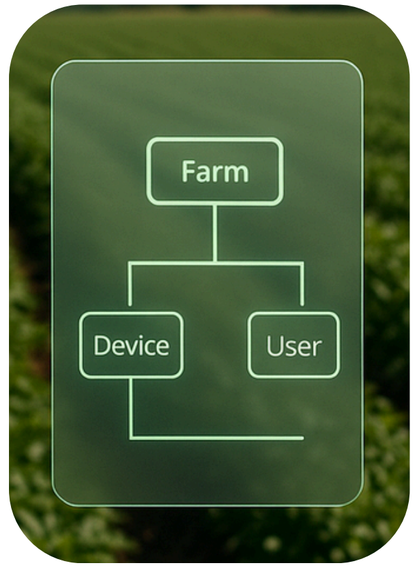
Topics

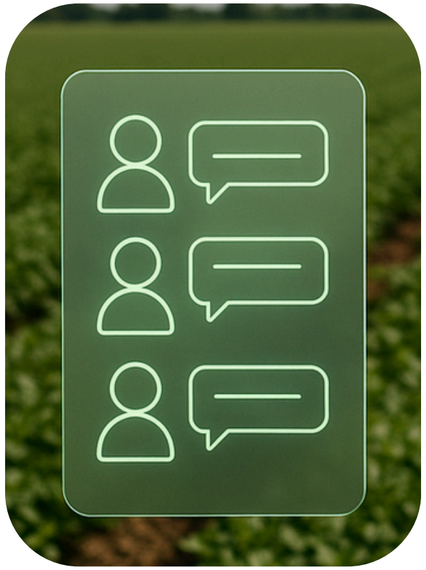
1. introduction
2. Vision
3. Design
4. Implementation
5. Team & Workflow
6. Beyond the Build

System Design Process

To ensure clarity, scalability, and feature alignment, our development process begins with structured system design artifacts:

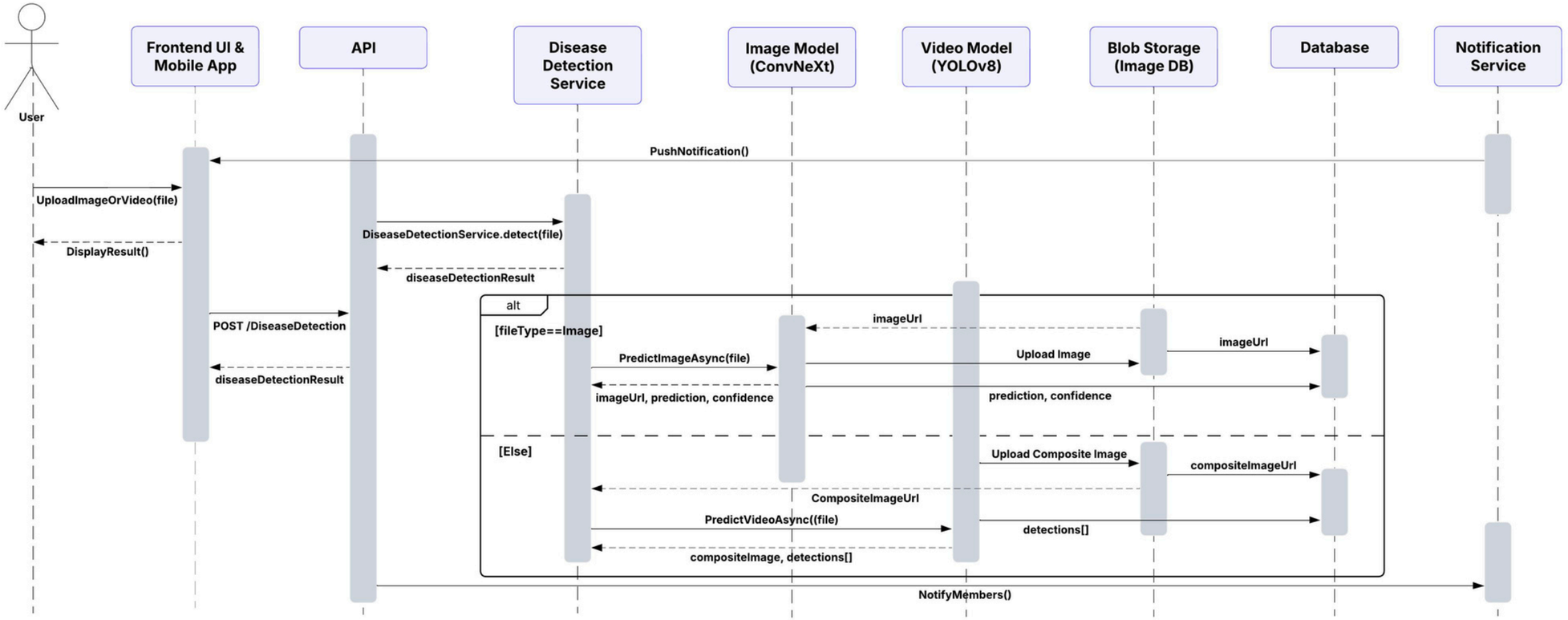
- **User Stories**
  - Captured real-world user needs and guided feature planning based on actual roles.
- **Use Case Diagrams**
  - Mapped the major system interaction for each role, visualizing user behavior and system responses.
- **Entity Relatioship Diagrams (ERDs)**
  - Modeled the database schema to represent farms, fields, devices, roles, and user interactions.
- **Feature Breakdown & Prioritization**
  - Translated the above into modular, implementable features – prioritized by impact and technical feasibility.



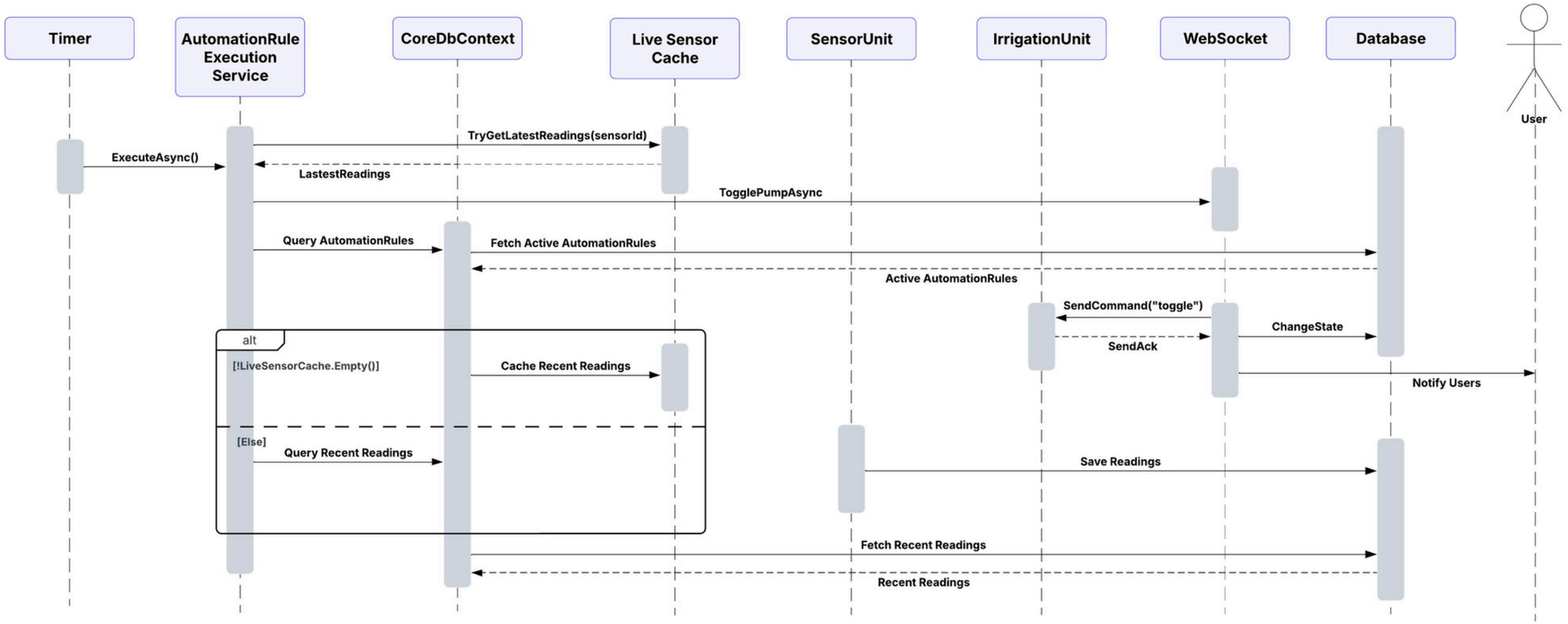


- **Use Case Diagram**

- **Disease Detection Sequence Diagram:**

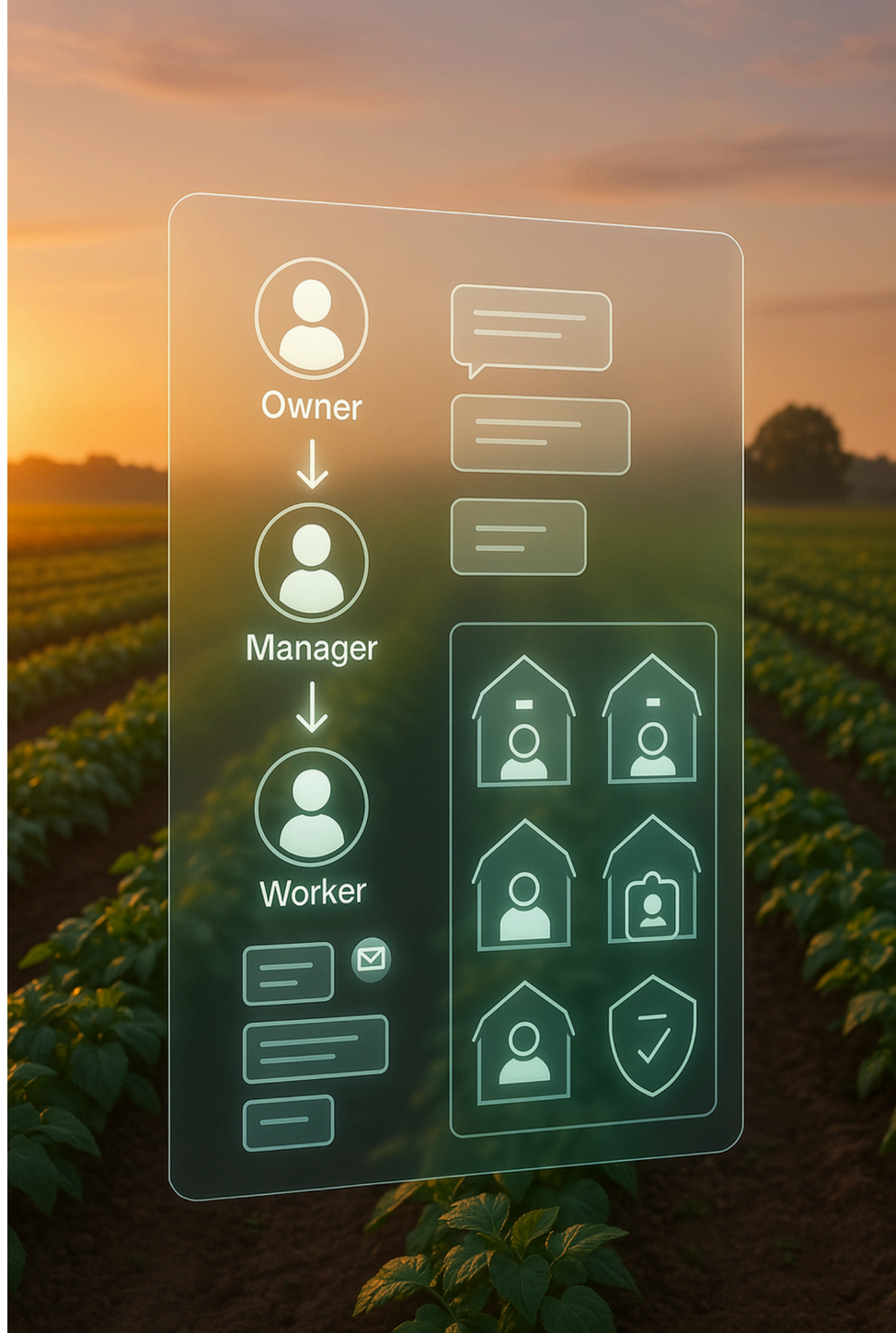


- **Automation Sequence Diagram:**



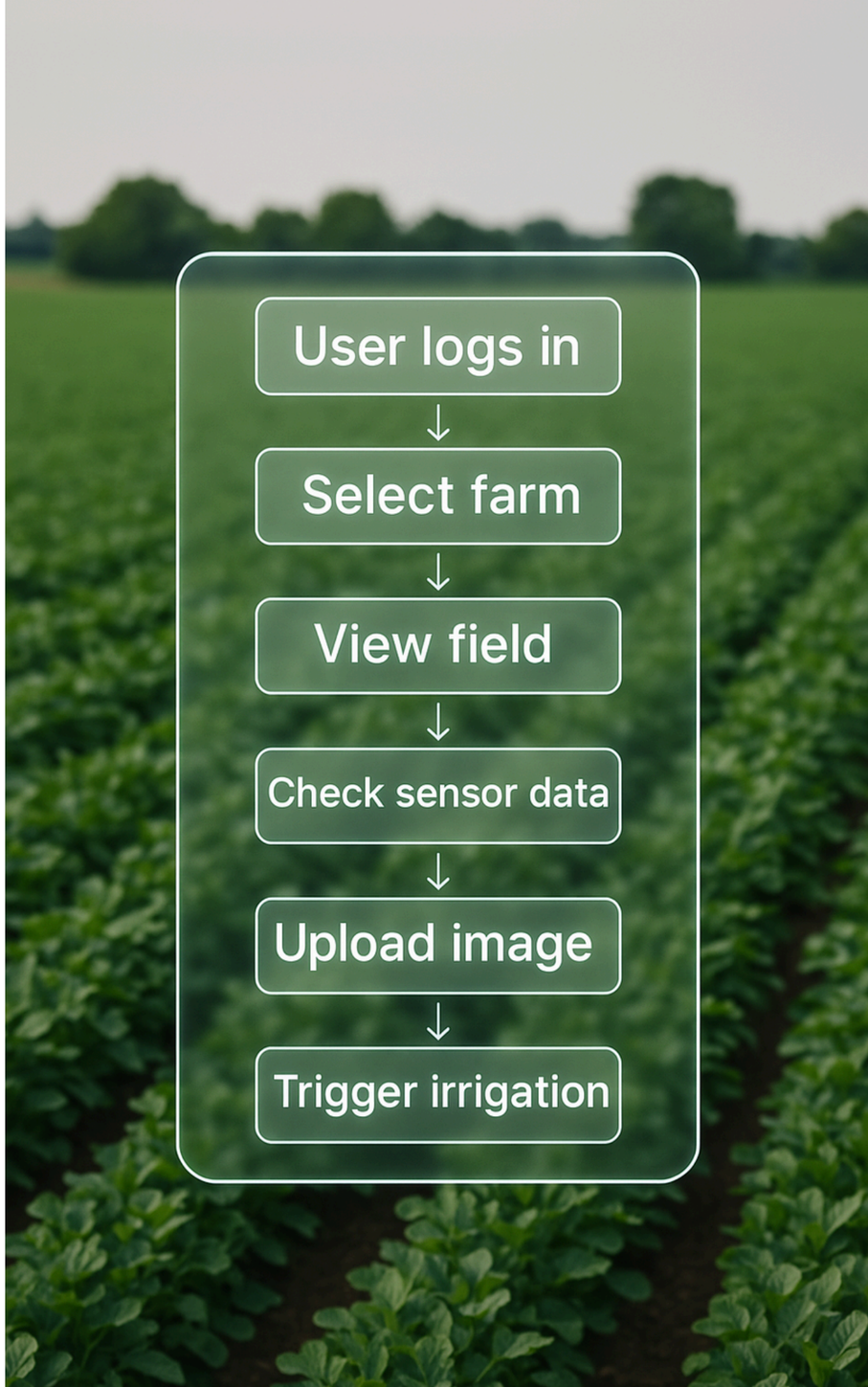
Role-Based Multi-Tenant Workflow

- **Roles**
  - Owner: full control, manages roles and farm profile.
  - Manager: assign tasks, manages the farm.
  - Expert: disease review, analytics.
  - Worker: task execution.
- **Multi-Tenant System Design**
  - Users can belong to multiple farms with distinct roles per farm.
  - Access control is scoped per farm context.
  - Custom-built identity and claims system ensures flexibility, security, and clarity.
- **Task Assignment & Workflow**
  - Owners → Managers → Workers
  - Task status updates in real-time with alrert in app + via email.



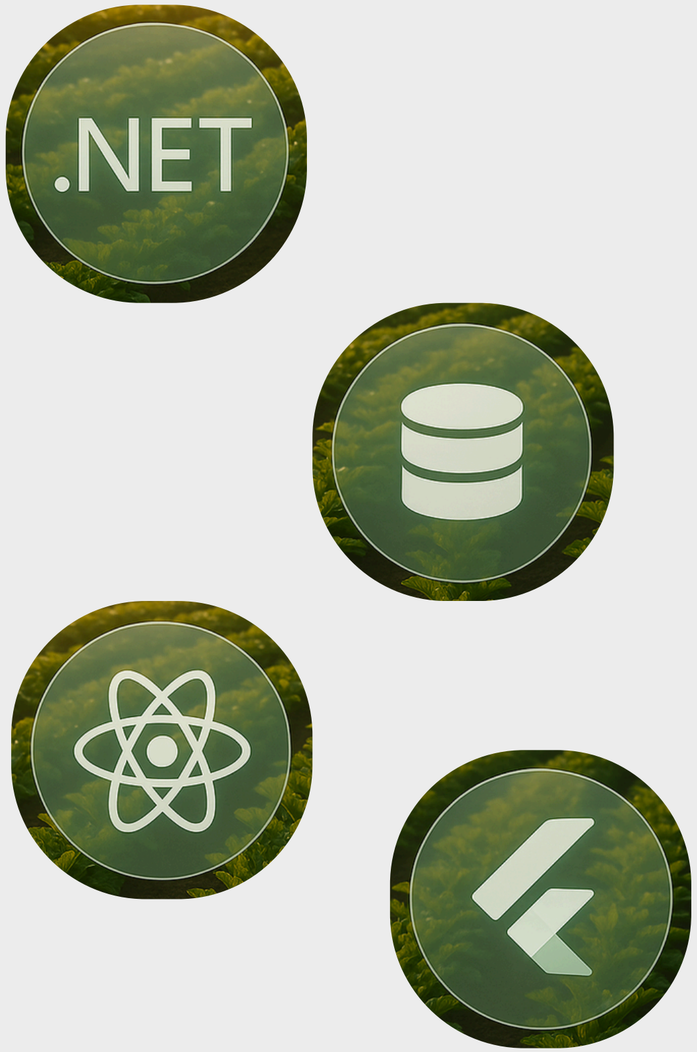
User-Journey – From Login to Action

- **Step 1: Secure login**
  - User authenticates using email/password and have a JWT issued.
- **Step 2: Farm Selection**
  - Multi-Tenant view displays all user-associated farms with the dashboard loaded based on selected farm.
- **Step 3: Field Overview**
  - Field cards with visual health indicators, recent activity logs, and alerts.
- **Step 4: Real-Time Data Monitoring**
  - Live soil moisture, temperature, and humidity readings.
- **Step 5: AI-Driven Crop Health Check**
  - User uploads plant image or drone footage and the model classifies the disease with detection history available per field.
- **Step 6: Smart Irrigation Trigger**
  - Based on sensor thresholds or manual input irrigation is triggered and or activated remotely.



Topics

1. introduction
2. Vision
3. Design
4. Implementation
5. Team & Workflow
6. Beyond the Build



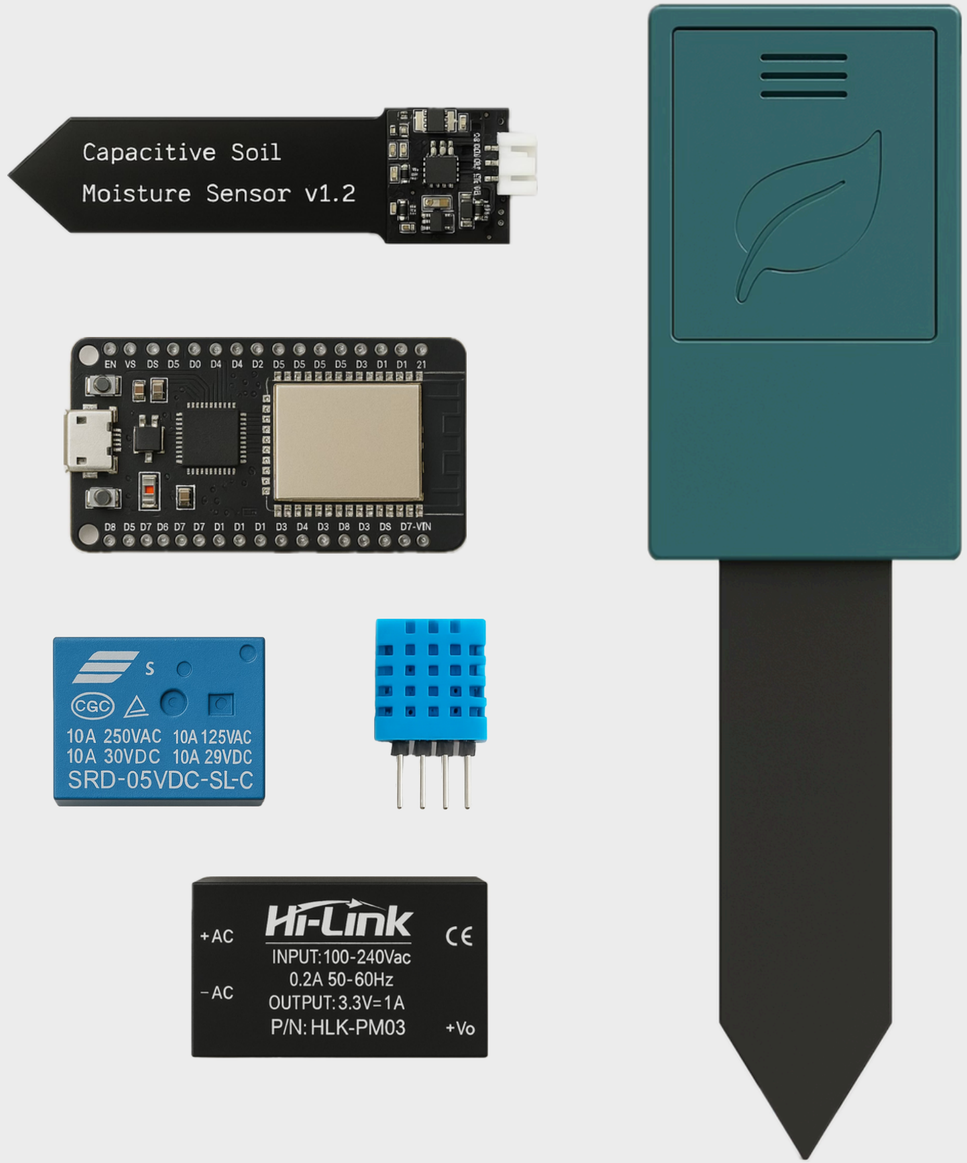
Tech Stack and Architectural Model

- **Tech Stack:**
  - .Net Core 9
  - React + Tailwind
  - Azure Sql Server
  - Tensorflow,PyTorch
  - Flutter

- **Clean Architecture:**
  - Domain (Entities, Value Objects)
  - Application (Use Cases, Interfaces)
  - Infrastructure (EF Core, AI, Auth)
  - Presentation (React, Web API)

IoT Hardware

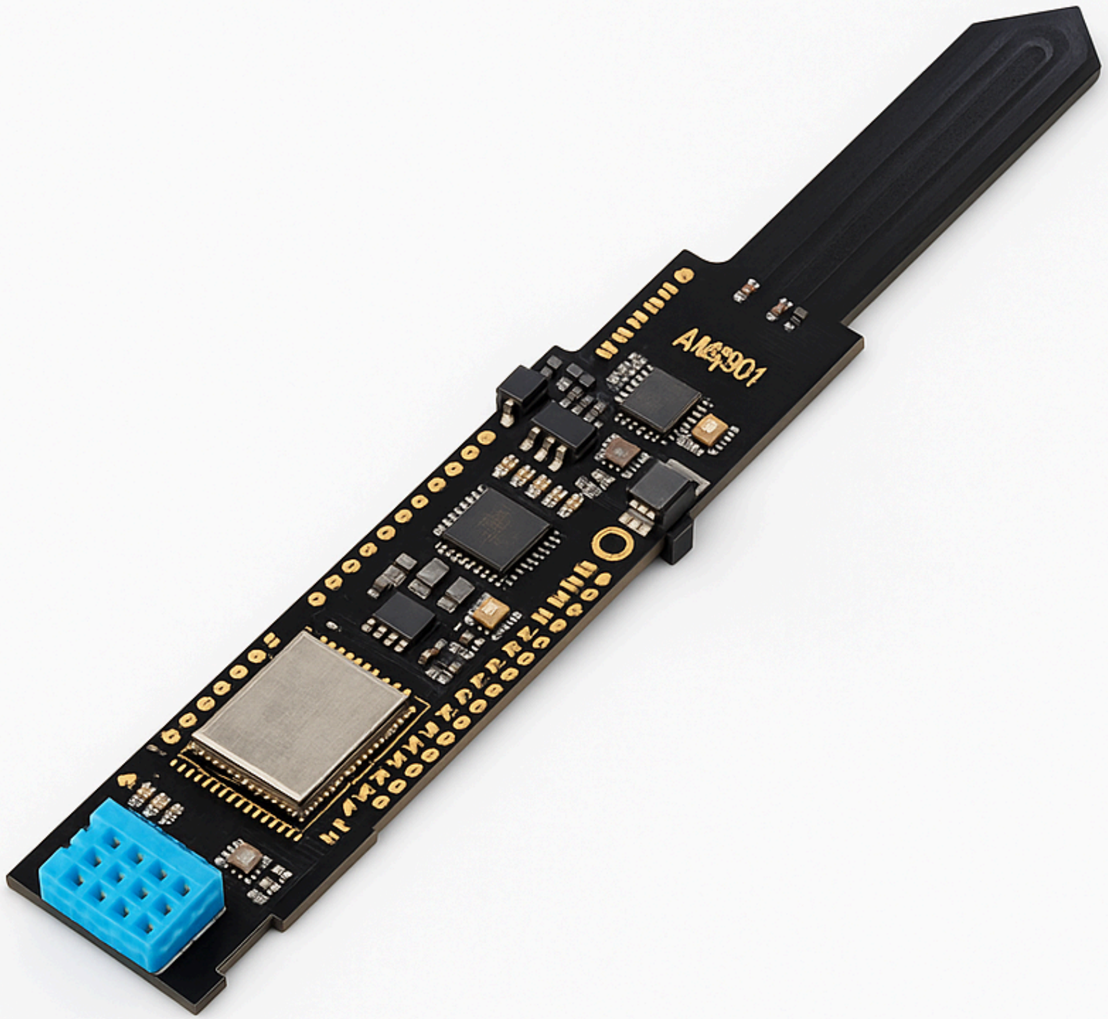
- **Sensor Unit:**
  - ESP32 WROOM
  - Temperature and humidity sensor
  - Moisture Sensor
  - LiPo battery
- **Irrigation Unit:**
  - ESP32 WROOM
  - Relay module (SRD-05VDC-SL-C)
  - AC/DC converter (HLK-PM03)



Custom sensor design

- To optimize performance and manufacturability, we plan to design and fabricate a custom PCB that integrates all necessary components into a single, compact board.

- **Benefits of Custom Design:**
  - Reduced size and enclosure complexity
  - Improved durability and weatherproofing
  - Lower production cost at scale
  - Seamless deployment across large farms



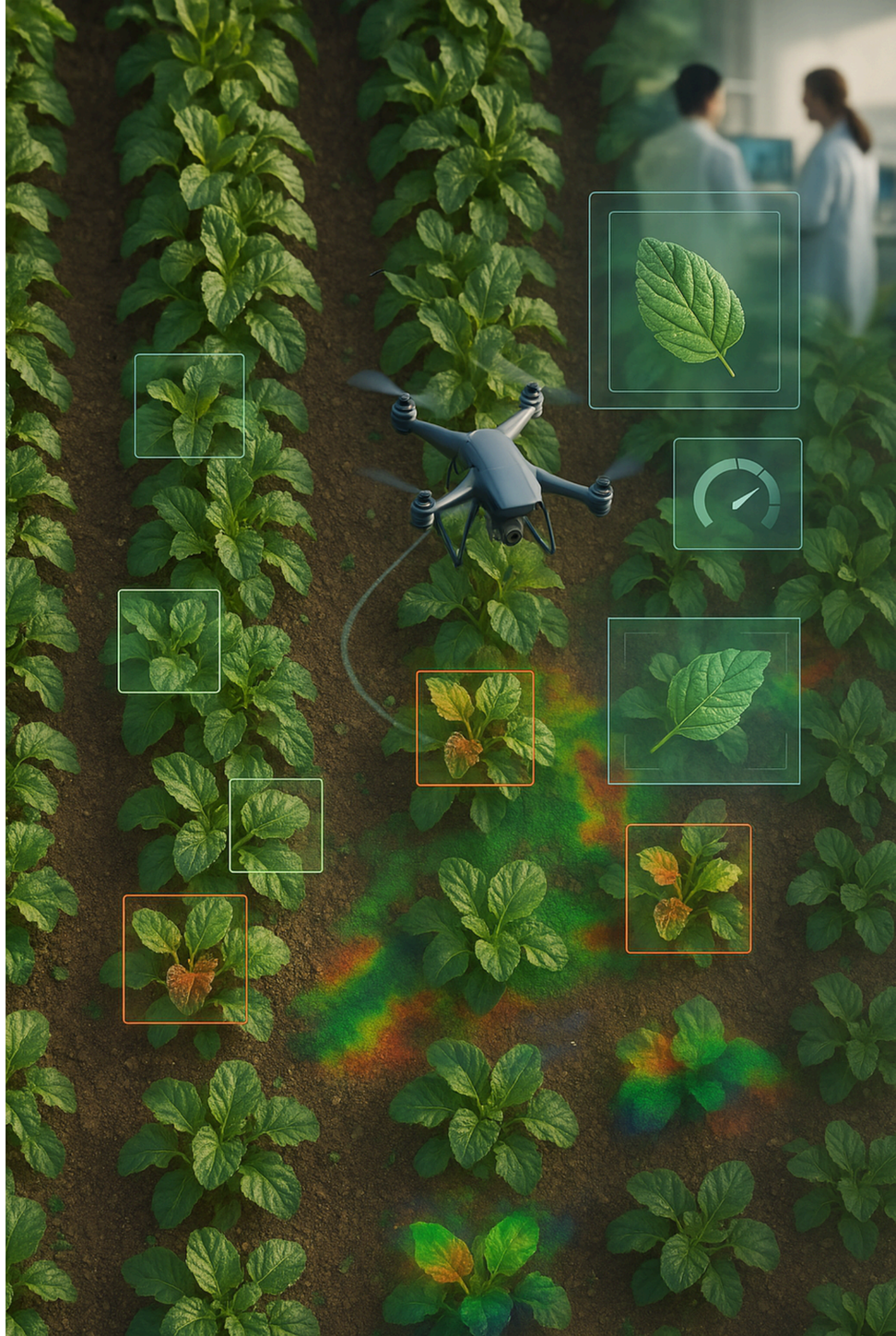
IoT Software and Connectivity

- **Software**
  - Custom lightweight firmware written in C++
  - Fail-safe mechanism with disconnection timeout.
  - Clear abstraction for sensor logic and error handling
  - Secure communication over Wi-Fi

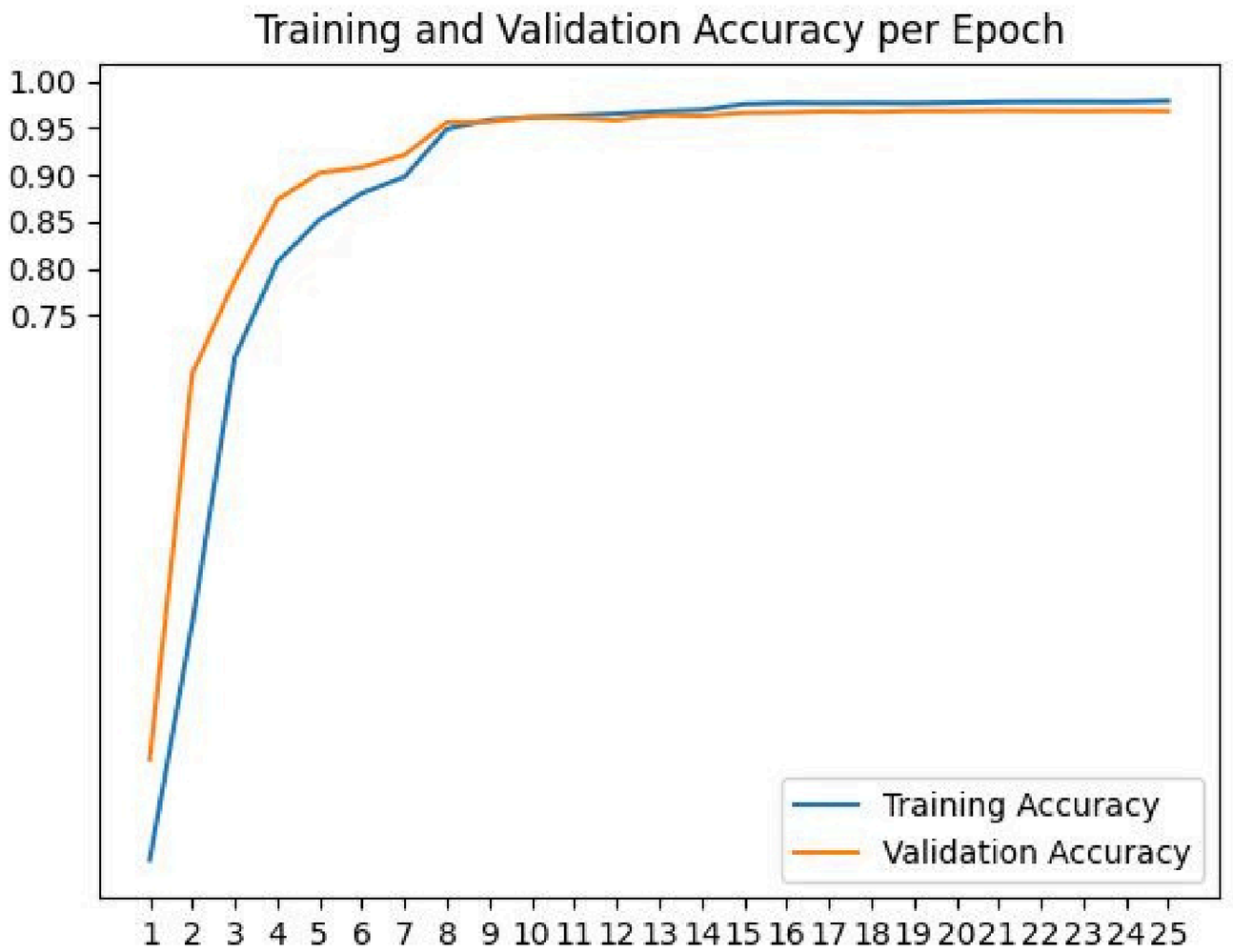
- **Connectivity**
  - Cloud sync via Wi-Fi (WebSocket))
  - Connection moniitoring via ping mechanism
    - The server sends a ping every 10 seconds.
    - Devices are expected to answer with pong.
    - After failing 6 pings the devices is labeled offline.

Disease Detection Models

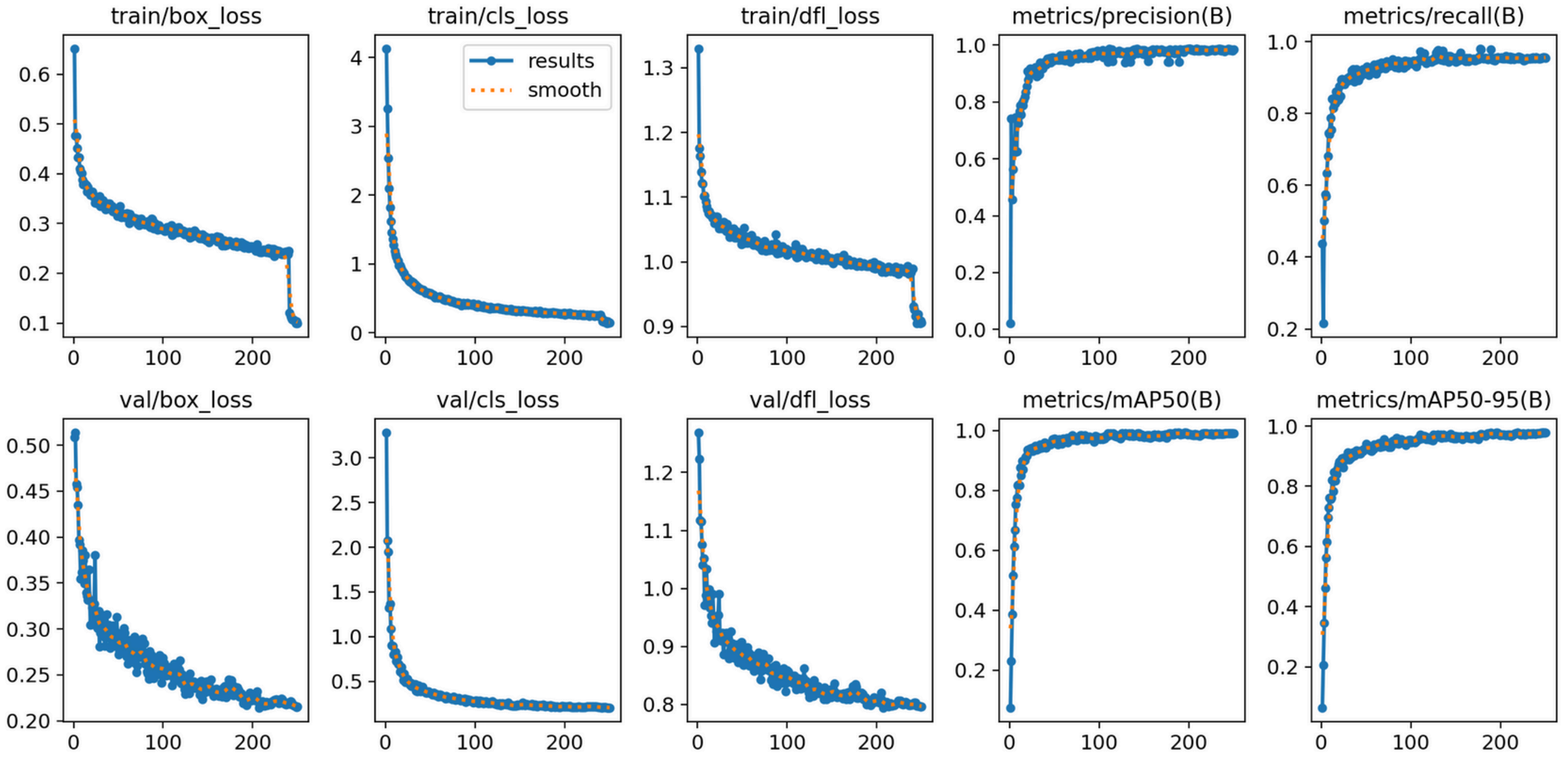
- **Models:**
  - YOLO for leaf detection (weeds or infected)
  - CNN for disease classification and treatment recommendation
  - Supports images + mid range drone video (up to 3m)
- **Training & Accuracy:**
  - Dataset: Over 54,000 datamined from diverse sources.
  - Used for training a ConvNeXt model to classify plant diseases.
  - Augmentation techniques: brightness, rotation, blur, angle variation.
  - Currently collecting and datamining datasets for YOLO-based object detection.
  - Results: accu.
- **Model Restrictions**
  - User selects supported crop on setup to ensure model relevance and accuracy.
  - Currently supports 14+ crops with trained disease model.



ConvNeXt


Training and Validation Accuracy per Epoch
1.00
0.95
0.90
0.85
0.80
0.75
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Training Accuracy
Validation Accuracy

YOLOv8


train/box_loss
train/cls_loss
results
smooth
train/dfl_loss
metrics/precision(B)
metrics/recall(B)
val/box_loss
val/cls_loss
val/dfl_loss
metrics/mAP50(B)
metrics/mAP50-95(B)

Backend Implementation

Why is the Backend Important?

- The backend powers the core logic, data processing, and system integrations behind the user interface.

It ensures:

- Business Logic Execution
- Data Management
- Scalability & Security

Clean Architecture

We adopted Clean Architecture to build a more robust and organized backend structure.

Key Benefits:

- Separation of Concerns
- Testability
- Flexibility & Maintainability

Backend Implementation

Why We Used .NET Core 9?

.NET Core 9 was chosen for its power, flexibility, and compatibility with m
software needs.

Key Benefits:

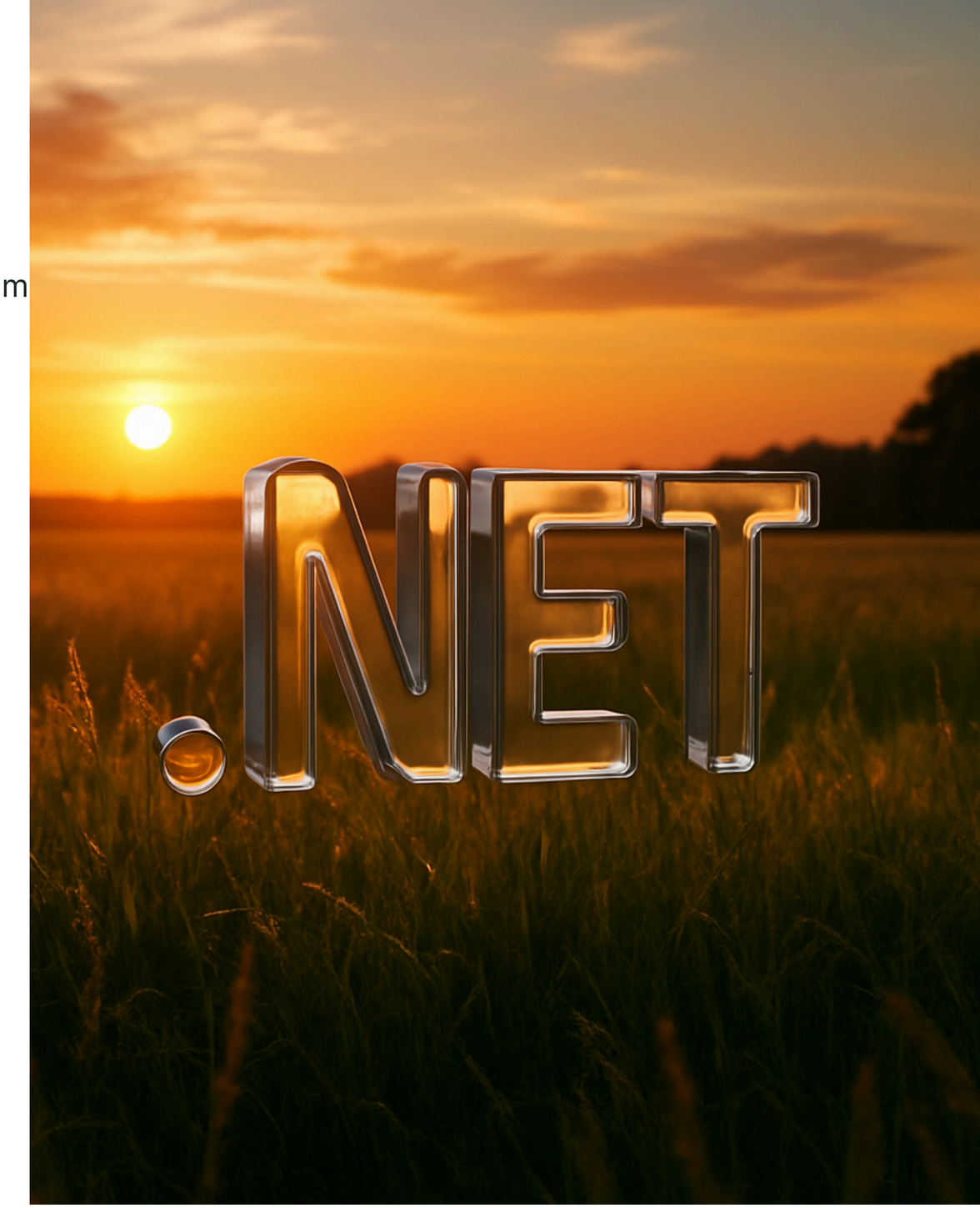
- Cross-Platform
- High Performance
- Rich Ecosystem
- Cloud Ready

Backend Components

Our backend follows a clean, layered architecture based on .NET Core.

Component Layers:

- Domain Layer
- Application Layer
- Infrastructure Layer
- Presentation Layer



Frontend (Web) Implementation

Why is the Web Important?

- The web is a vital part of modern life and a core platform for delivering services across many sectors: education, commerce, healthcare, and agriculture.
- Through the web, systems can offer:
  - Anywhere, anytime access via any standard browser.
  - A flexible and seamless user experience without the need for software installation.
  - The ability to support multiple users simultaneously in a centralized and secure manner.
  - Instant updates and maintenance without requiring action on each individual device.
- Therefore, developing a web interface is a key step in any software project to ensure scalability, efficiency, and usability, especially in fields that require ongoing interaction between the user and the system—such as modern agriculture.

Frontend (Web) Implementation

Brief Overview of React

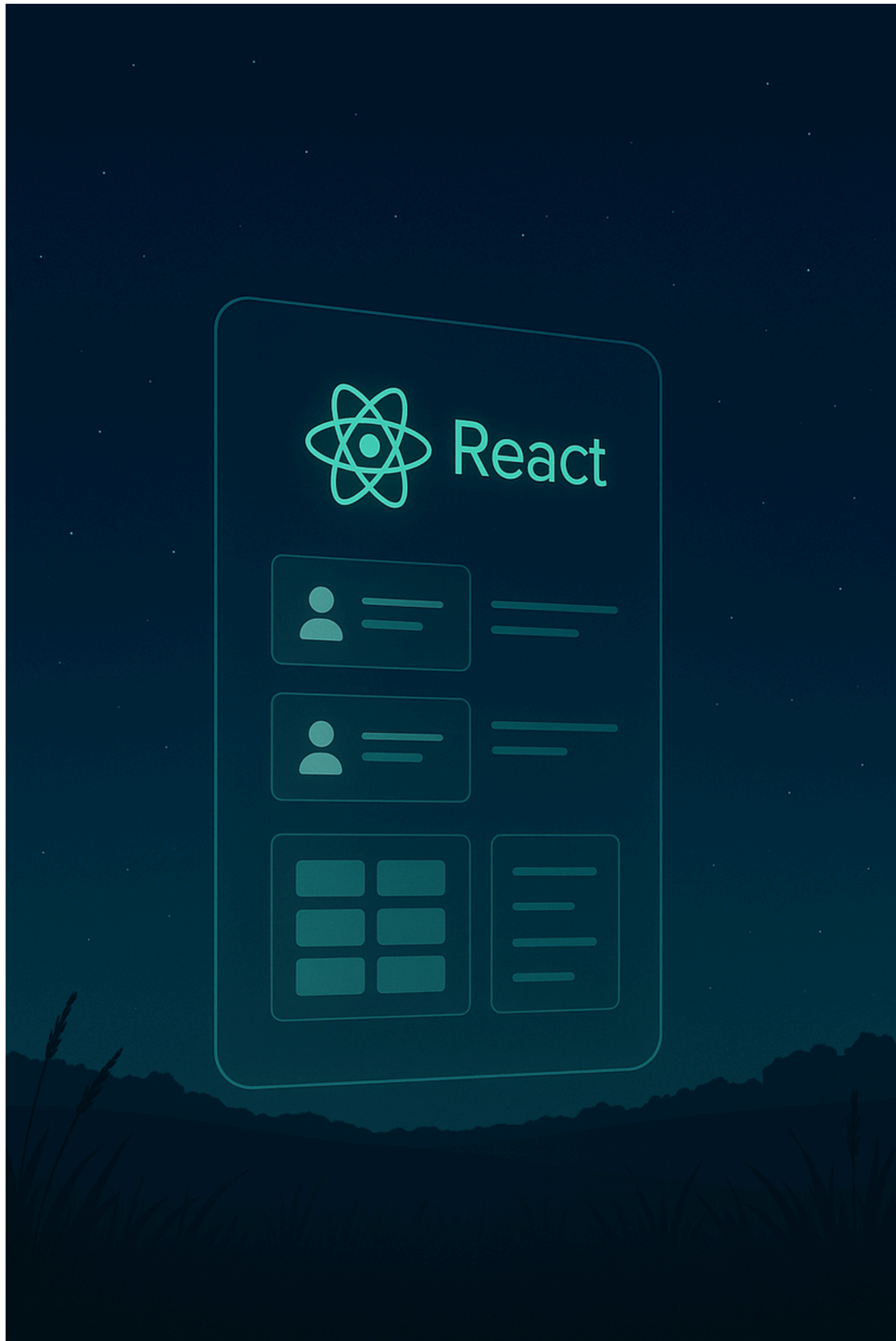
- React is an open-source JavaScript library developed by Meta (Facebook) for building fast, dynamic, and interactive user interfaces—especially single-page applications (SPAs).

Why We Used React in Our Project?

- Component-Based Architecture: Makes the code modular and reusable.
- High Performance: Uses a virtual DOM for faster updates and rendering.
- Responsive Design: Easily adapts to different screen sizes and devices.

Why We Chose React Instead of Angular?

- React is easier and faster to learn and implement, which fits well with the limited time frame of a graduation project.
- It offers greater flexibility in choosing tools and libraries.
- React uses plain JavaScript, which we are already familiar with, unlike Angular which requires TypeScript.



Frontend (Web) Implementation

Brief Overview of Tailwind CSS

- Tailwind CSS is a utility-first CSS framework used to design responsive and fully customizable user interfaces with ease.

Main Advantages:

- Faster Development

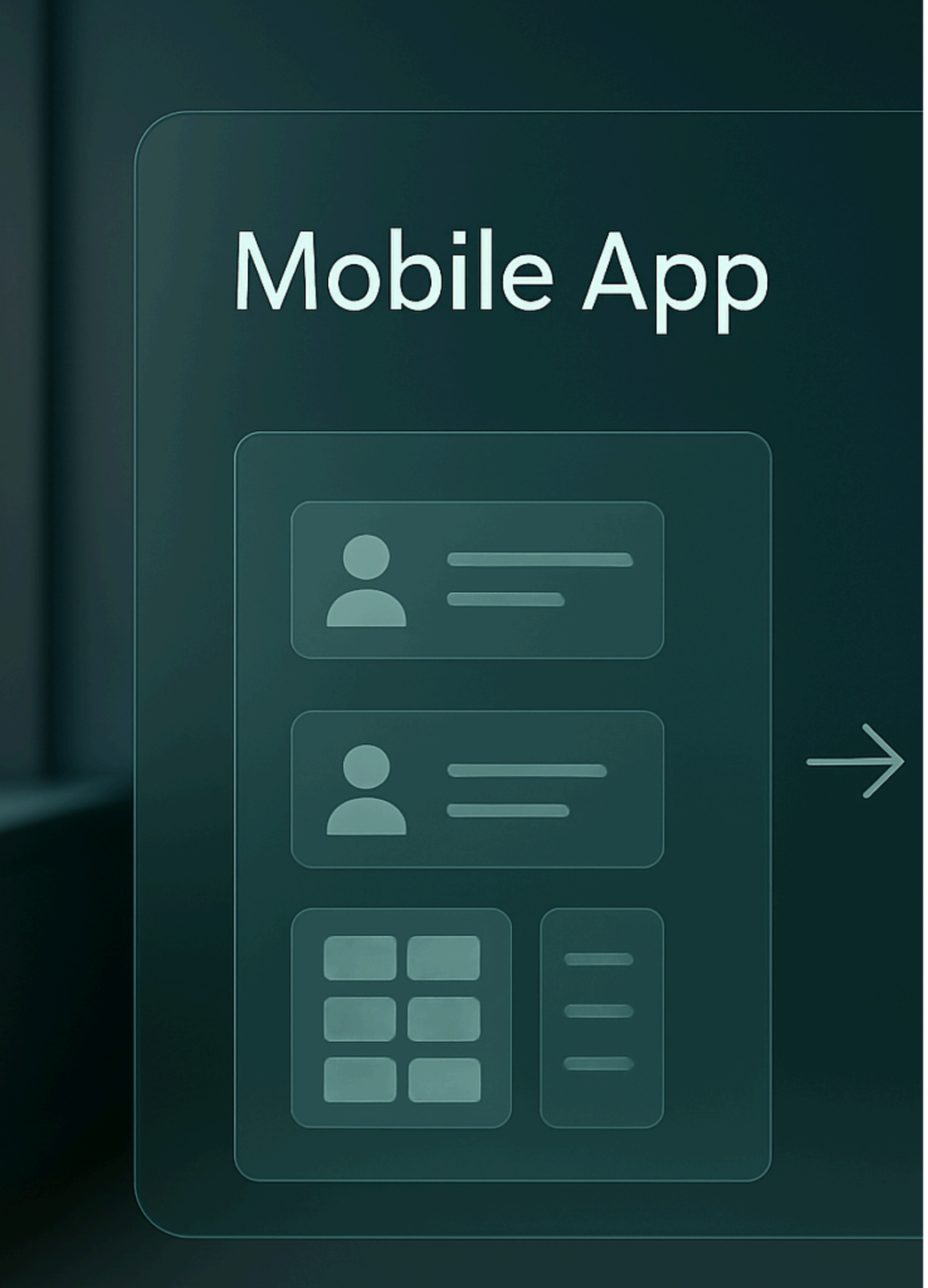
  Write most styles directly in the HTML without needing separate CSS files.

- Responsive Design

  Built-in support for multiple screen sizes using simple utility classes.

- Full Customization

  Easily customize everything from colors and spacing to typography and layout.



Mobile Implementation

Why is the Web Important?

cross-platform mobile app developed using Flutter, aiming to digitize and enhance smart agriculture with modern features and real-time capabilities follows

Why Flutter Was Chosen for Agrivision ?

- Single Codebase
- Declarative UI
- Hot Reload
- Direct native access
- Rich Plugin Ecosystem

a modular architecture to ensure scalability, clean code, and maintainability.

Main Project Folders:

- ·Screens: Widgets:
- ·Bloc & Cubit:
- ·Models:
- ·Services:
- ·Components:
- ·Assets:

Mobile Implementation

Core Services:

- **REST API Service:**

  Centralized service (ApiService) for sending HTTP requests (GET/POST) to the backend Used for:

  - ·User registration/authentication
  - ·Fetching data (farms, devices, etc.)
  - ·Sending/receiving messages via REST

- **SignalR Hub Service:**

  Real-time communication layer using SignalR. Enables live updates for:

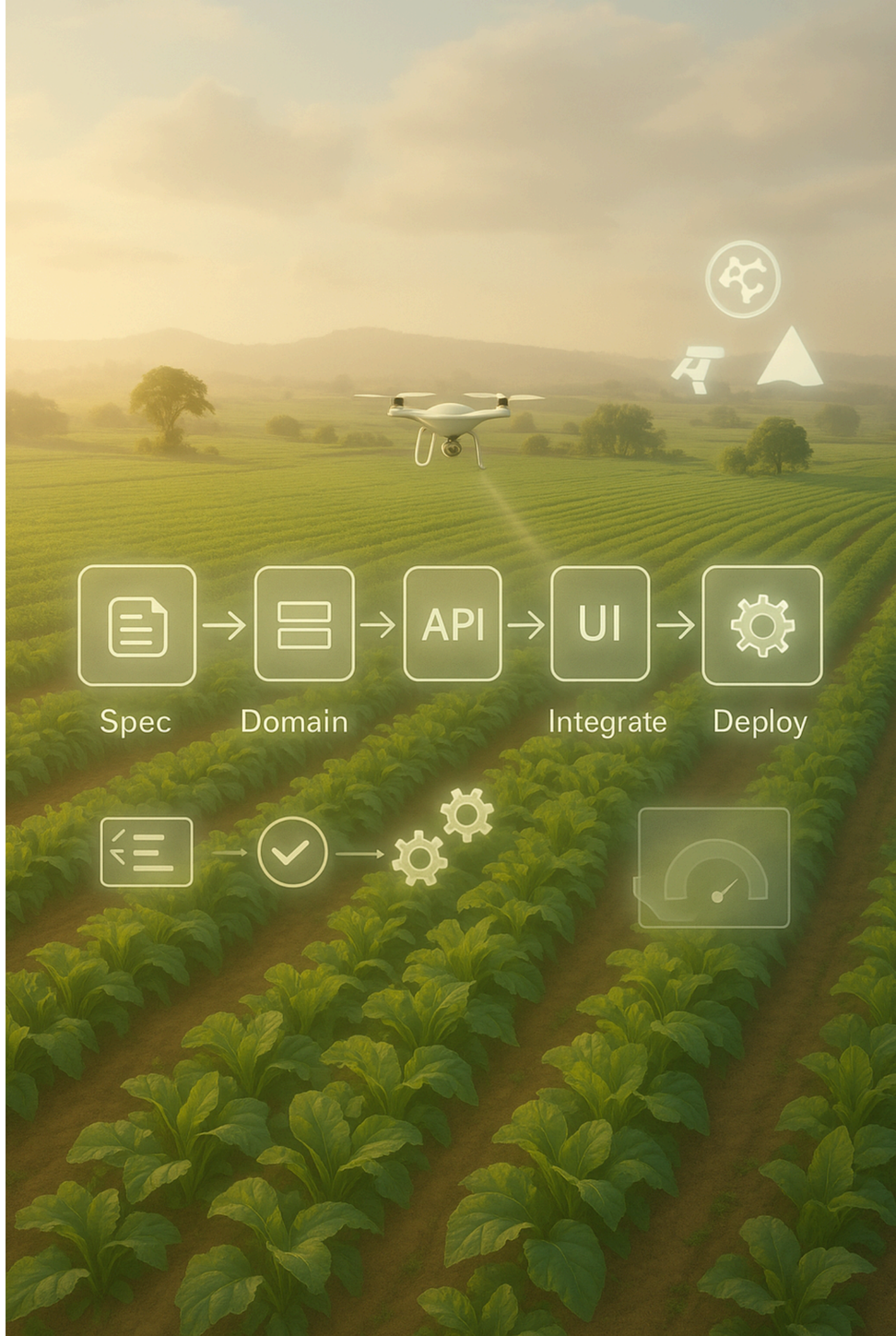
  - ·Sensor data streams (temperature, humidity, etc.)
  - ·Real-time messaging in chat features

Topics

1. introduction
2. Vision
3. Design
4. Implementation
5. Team & Workflow
6. Beyond the Build

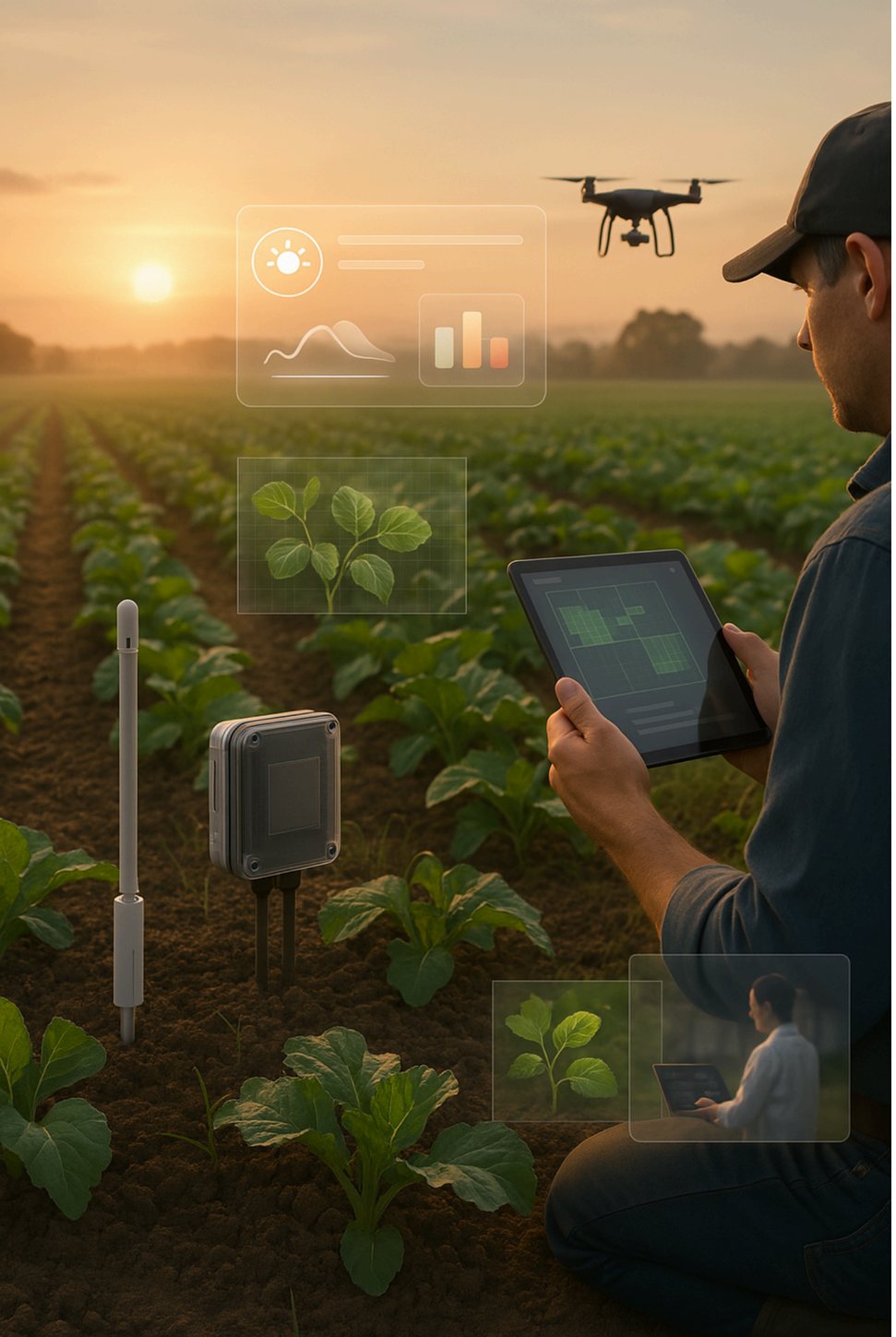
Workflow Management

- **Methodology:**
  - a CI/CD-driven feature streaming agile approach.
  - Weekly standup, flexible sprint cycles, and asynchronous collaboration.
- **Development Lifecycle Flow:**
  - Spec → Domain → API → UI → Integrate → Deploy.
  - Provides clarity, modularity, testability.
- **CI/CD & DevOps:**
  - Github
  - Azure Pipelines
  - Vercel for frontend auto-deploy



Topics

1. introduction
2. Vision
3. Design
4. Implementation
5. Team & Workflow
6. Beyond the Build



Key Challenges & Real World Testing

- **Key Challenges:**
  - ESP32 field stability.
  - Real-time updates under flaky Wi-Fi.
  - Increasing model accuracy and generalization.
  - Cost of cloud resources.
  - Security and access control

- **Real World Testing:**
  - Coordinated with a local farm owner to test agrivision in a real agricultural setting.
  - Done field deployment scheduled for mid-June.
  - Goals: validate system reliability, gather feedback on usability, test AI performance in real-world conditions
  - Includes full deployment: sensors, irrigation, dashboard, and disease detection.

Image From Real World Testing

Image From Real World Testing

Future Enhancements

Hardware Reliability: Replace basic sensors like DHT11 with industrial-grade

- **On-Device Intelligence:** Deploy lightweight ML models
- **Smarter Automation:** Extend rule logic to consider weather data, crop-specific thresholds, and historical soil conditions for adaptive control.
- **OTA Updates:** Implement secure over-the-air firmware updates for remote maintenance
- **ESP-NOW Fallback:** Use ESP-NOW for peer-to-peer communication

References

1. Food and Agriculture Organization (FAO), "The impact of plant diseases on food security," www.fao.org
2. United Nations Water, "Water use in agriculture," www.unwater.org
3. IPCC Special Report on Climate Change and Land, www.ipcc.ch
4. McKinsey & Company, "Agriculture's connected future: How technology can yield new growth," www.mckinsey.com

Thank You

Team Members

Youssef Mahmoud Mohamed

Hussein Mohamed Al-Shahat

Abdelrahman Maher Abdo Ali

Shehab Ahmed Mohamed Ahmed

Mohamed Omar Saber

Nada Yousry Mohamed

Ahmed Yousry Mahmoud