Technical Requirements Document for the “AgroBoost” Project

1. **Full name of the work: “AgroBoost”**

2. **Team name and technology stack used**

Our team named “Al Xorazmiy”. Project employs Django for backend operations, Vue.js for frontend interfaces, and Kubernetes for scalable deployment, with a primary focus on Python as the core programming language.

✓ ***Backend Framework***: Django (Python)

Core Responsibilities:

- Serve RESTful API endpoints.
- Manage data storage, retrieval, and processing of analytical results from ML models.
- Integrate with machine learning services for image/video analysis.
- Implement user authentication and role-based access controls.

✓ ***Frontend Framework***: Vue.js

Core Responsibilities:

- Interface with the backend APIs to present analysis results.
- Provide an interactive, responsive dashboard for data visualization and user interaction.
- Facilitate functionalities such as video/image uploads, field monitoring, and reporting.

✓ ***Machine Learning Module Technologies***: OpenCV, Python

Core Responsibilities:

- Image and video analysis to detect plant health status, diseases, pests, and nutrient deficiencies.
- Generate quantitative metrics such as NDVI (Normalized Difference Vegetation Index) and other vegetation indices.

- Suggest appropriate interventions based on detected issues.

✓ ***Deployment and Orchestration Platform***: Kubernetes

Responsibilities:

- Manage containerized deployments of backend, frontend, and ML modules.
- Ensure system scalability, high availability, and automated fault tolerance.

✓ ***Drone Integration*** **Technologies**: DJI SDK/MAVSDK, Python

Responsibilities:

- Capture high-resolution imagery and transmit data to the backend.
- Support real-time data streaming and ensure seamless integration with ML processing workflows.

3. **Project Assignment**

The **"AgroBoost"** project aims to deliver an advanced agricultural monitoring solution utilizing drone imagery, machine learning (ML), and computer vision technologies. The solution enables comprehensive analysis of farmland, offering actionable insights on crop health, pest identification, and nutrient deficiencies.

Our project offers various functionalities by client type:

For farmers:

- Get crop area information through an easy interface.
- Monitoring the condition of crops in real time.
- Receive automatic alerts for fertilizer and pest control measures.

For customers:

- Order food online.
- Product delivery service directly from farms.

For operators:

- Analysis of data from satellites and drones.
- Configure automated flight and monitoring plans for drones.

4. **Functional Requirements**

✓ ***Image/Video Analysis:***

- **Data Capture**: Drones equipped with high-resolution cameras should autonomously capture farmland imagery;
- **Health Assessment**: Analyze plant health using vegetation indices and ML models to provide greenness measurements;
- **Disease & Pest Identification**: Employ computer vision techniques to detect diseases, pests, and other anomalies.
- **Nutrient Analysis**: Analyze leaf coloration to detect deficiencies and provide actionable nutrient management recommendations.
- **Pesticide Recommendations**: Suggest targeted interventions based on pest/disease identification.

4.1. **User Interface**

- **Dashboard Visualization**: Display a comprehensive view of scanned fields with health indicators using geospatial mapping techniques.

- **Data Reporting**: Offer detailed reports and analytics on crop health, diseases, and suggested treatments, accessible via the user dashboard.

- **Alert Mechanism**: Real-time alerts for detected issues, tailored to user roles (e.g., farmers, agronomists).

4.2. **Data Management**

- **Storage:** Use a structured database system for storing drone footage, analysis results, and historical data.

- **Retrieval and Search**: Implement advanced filtering and querying capabilities based on parameters such as date, geographic location, or crop type.

4.3. **User Authentication and Access Management**

Authentication Mechanism: Implement secure user authentication using JWT tokens and Django's built-in authentication modules.

Role-Based Access Control: Differentiate access and functionalities based on predefined user roles.

5. Performance and Scalability

Throughput: The system must handle multiple concurrent drone uploads and processing requests efficiently.

Latency: Ensure analysis results are delivered with minimal delay, achieving near-real-time feedback.

5.2 Reliability and Availability

Redundancy: Utilize Kubernetes for failover and redundancy to ensure high availability.

Data Backup: Configure automated data backup strategies with appropriate retention policies.