1. Backend

Component	Choice	Notes
Language	**Node.js**	Fast, async, integrates well with frontend; good for API orchestration
Framework	**Express.js**	Lightweight, easy to set up REST APIs
AI/ML	**External APIs** (Plant.id, AgroAI, Microsoft Custom Vision, IBM Watson Agriculture)	No need to use PyTorch/TensorFlow locally
Sensor/IoT	**APIs from sensor providers** (Libelium, Adafruit, Particle.io)	Fetch soil moisture, temperature, humidity, leaf wetness
Database	**MongoDB**	Flexible storage for images, sensor readings, prediction logs
Task Queue	**Optional:** Node Cron or Bull.js	Schedule API calls, batch processing, alerts
Notifications	**Firebase Cloud Messaging / OneSignal**	Mobile-friendly alerts to farmers

2. Frontend

Component	Choice	Notes
Framework	**React.js**	Team already knows this; supports modern interactive dashboards
Styling	**Tailwind CSS / Material-UI**	Responsive, prebuilt UI components
Charts / Maps	**Plotly.js / Chart.js / Leaflet.js**	Trend plots, spectral health maps, risk zones
API calls	**Axios / Fetch API**	Connect frontend to Node.js backend

3. Data & File Storage

Component	Choice	Notes
Image Storage	**AWS S3 / Google Cloud Storage**	Store raw multispectral/hyperspectral images
Database	**MongoDB**	Flexible, handles sensor data, predictions, user info

---

4. Optional / Future Enhancements

Component	Choice	Notes
Local AI / ML models	Python + FastAPI + TensorFlow / PyTorch	Only if you want custom AI models in future
Relational Database	PostgreSQL	Optional if you need structured farm/user management

---

5. Architecture Overview (High-Level)

1. **Frontend (React)**
    - Dashboard, plots, alerts, maps
2. **Backend (Node.js + Express)**
    - Handles API orchestration
    - Fetches AI predictions from external APIs
    - Fetches sensor data from IoT APIs
    - Fuses data and generates alerts
    - Stores everything in MongoDB
3. **External APIs**
    - AI/ML for crop stress/disease/pest detection
    - Sensor data for soil/temperature/humidity/leaf wetness
4. **Notifications**

    - Mobile alerts via Firebase / OneSignal
5. **File Storage**
    - Images on AWS S3 / GCP
6. **Optional Python microservice**
    - For future custom AI or hyperspectral processing

✅

Why this is best

- Leverages your **team's frontend strength**
- Uses **APIs for AI + IoT**, minimizing backend coding complexity
- Node.js + Express handles all **data orchestration & API integration**
- **Scalable and hackathon-ready** — focus on dashboard, alerts, and usability
- Minimal setup, fast development, and cloud-ready
