Technical Overview – System Architecture

System Architecture

- **Data Layer:**
  Synthetic soil, rainfall, NDVI, and fertiliser datasets stored as CSV/GeoJSON in data/.
  Derived feature table (panels.csv) used for model training.
- **Model Layer:**
  scikit-learn Histogram-Based Gradient Boosting Classifier (HGB).
  Features: soil nutrients, NDVI trends, rainfall, fertiliser load, slope/flow proxies.
  Target: synthetic risk_label generated by heuristic risk function.
- **API Layer:**
  FastAPI REST endpoint /score — serves model predictions and generates Alert Card JSONs.
- **Dashboard Layer:**
  Streamlit front-end visualising per-field risk status (Green / Amber / Red) and top feature drivers.

Training & Validation

- Training set: 75%, test set: 25% split.
- Model metrics:
  - AUC = 0.998
  - Accuracy = 0.995
  - Precision / Recall / F1 = 0.667 / 0.667 / 0.667
- Feature importance: nitrogen load > recent rainfall > NDVI average.

Technology Stack

- Python 3.11, Pandas 2.2, NumPy 2.0, scikit-learn 1.5
- FastAPI 0.115, Uvicorn 0.30, Streamlit 1.38
- IDE: VS Code (Windows 10 environment)

File Structure

```
Agric Project/
│
├── data/            # synthetic dataset package
```

```
├── outputs/          # model + metrics + plots
├── generate_data.py    # Step 1: dataset generator
├── train_model.py      # Step 2: model training
├── serve_api.py        # Step 3: alert card API
├── dashboard.py        # Step 4: Streamlit dashboard
└── requirements.txt
```