Planning Logic

🔹 1. Project Initialization

- Project Title: AI/ML-Based Detection of Rotten Fruits and Vegetables
- Project Type: Computer Vision using AI/ML
- Stakeholders: Farmers, Supply Chain Operators, Retailers
- Team Setup: Data Scientist, ML Engineer, Software Developer, Domain Expert

---

🔹 2. Problem Definition

- Problem Statement:
  Significant post-harvest losses occur due to inefficient detection of rotten produce. Manual inspection is slow, costly, and inconsistent.
- Objective:
  Build an AI/ML system that can accurately and efficiently identify rotten fruits/vegetables using image data.

🔹 3. Requirement Gathering

- **Functional Requirements:**
  - Upload or capture images of produce
  - Detect and classify as “Fresh” or “Rotten”

- Generate real-time output or alerts

- **Non-Functional Requirements:**
    - High accuracy (>90%)
    - Fast inference (<1 second)
    - Scalable and user-friendly system

🔹 **4. Dataset Planning**

- ## Data Sources:
    - Public datasets (e.g., Kaggle, Fruit360)
    - Custom dataset collection via camera

- ## Data Characteristics:
    - Different fruit/vegetable types
    - Varying lighting and angles
    - Multiple stages of spoilage

- ## Data Annotation:
    - Manually label images as "Fresh" or "Rotten"
    - Store metadata (type, time, location)

🔹 5. Data Preprocessing

- Resize images to uniform size (e.g., 224x224)
- Normalize pixel values (0–1 range)
- **Data augmentation:**
    - Flip, rotate, blur, brightness adjustment
- **Split dataset:**
    - Training (70%) | Validation (15%) | Test (15%)

🔹 6. Model Selection

Choose suitable ML/DL models:

Type	Model	Use Case
CNN	Custom CNN	Lightweight, fast classification
Transfer Learning	MobileNet, ResNet	Accurate and efficient
Object Detection	YOLO, SSD	Detect location of rot on fruit

🔹 7. Model Training

- Train model using annotated dataset
- Monitor metrics: Accuracy, Precision, Recall, F1-score
- Tune hyperparameters (batch size, learning rate, epochs)
- Use tools like TensorFlow/Keras or PyTorch

🔹 8. Model Evaluation

- Use confusion matrix for error analysis
- Evaluate on test set and real-world images
- Measure:
    - Accuracy (>90%)
    - Latency (ms)
    - False positives/negatives

🔹 9. System Integration

- Front-End: Mobile app or web portal
- Back-End: Flask API or FastAPI for model serving
- Edge Deployment (optional): TensorFlow Lite on Raspberry Pi

---

◆ **10. Testing Phase**

- Functional Testing: Is output correct for known inputs?
- Performance Testing: Is it fast and reliable?
- UAT (User Acceptance Testing): Real users test the system in the field

---

◆ **11. Deployment**

- Deploy model on cloud/server/mobile
- Host API using Flask/Django
- Deploy dashboard for monitoring predictions and performance

---

◆ **12. Maintenance & Retraining**

- Regularly update dataset with new produce images
- Retrain model periodically
- Collect feedback from users to improve accuracy

◆ **13. Documentation & Reporting**

- **Prepare final project report**
- Include:

- Architecture diagram
- Dataset description
- Model performance
- Screenshots