**Project Design Phase**

**Solution Architecture**

|  |  |
| --- | --- |
| Date | 19 February 2026 |
| Team ID | LTVIP2026TMIDS77295 |
| Project Name | Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables. |

**Solution Architecture Diagram:**



*Figure 1: Architecture and Data Flow of the Smart Sorting Application (Fresh vs Rotten Detection)*

The diagram shows how your Smart Sorting system works end‑to‑end, from user upload to AI prediction and storage.

1. User Interface (Left side)

* Users (farmer, vendor, inspector, admin) use a Web App or Mobile App.
* Main action: Upload Image of a fruit or vegetable that needs to be checked.

This is the entry point where the quality‑checking task starts.

2. Application Layer – Online Prediction (Top block)

Inside the cloud platform (AWS / IBM Cloud / GCP):

1. Backend API (Python Flask)
   * Receives the uploaded image from the UI through an HTTP request.
   * Validates the file, saves it temporarily, and coordinates the next steps.
2. Image Processing
   * Converts the raw uploaded image into the format required by the model:
     + Resize (e.g., 224×224), normalize pixels, reshape tensor.
   * Produces a preprocessed image ready for inference.
3. Deep Learning Model (CNN)
   * Transfer‑learning model (e.g., MobileNetV2‑based) hosted in the application.
   * Takes the preprocessed image and predicts whether it is Fresh or Rotten.
   * The result is a class label plus probability/confidence.
4. Display Result (Fresh / Rotten)
   * Backend sends prediction back to the UI.
   * User sees: uploaded image + label (Fresh/Rotten) + confidence score.

On the right of this layer, supporting services are shown:

* Security – encryption and secure access for API calls and stored data.
* Scalability & Monitoring – auto‑scaling, health checks, metrics.
* Database – stores history, users, configuration.
* Storage – stores images, trained model files, logs.

3. Machine Learning & Admin Layer – Training and Management (Bottom block)

Also in the cloud platform:

* Deep Learning Model (CNN)
  + This is the training side. Dataset images are used with transfer learning to train or retrain the model.
  + Output is a trained model (weights file) used by the prediction API.
* Database (MySQL / PostgreSQL)
  + Stores meta‑data: image info, labels, training metrics, prediction logs.
* Storage (AWS S3 / IBM COS)
  + Stores large items: raw datasets, preprocessed images, model versions.
* Admin
  + Admin can manage dataset (upload new labeled images),
  + retrain the model, and
  + retain/roll back model versions.
* Infrastructure services
  + Load Balancer & Auto‑Scaling: distribute incoming requests across multiple app instances.
  + Kubernetes (optional): orchestrate containers for the web app and model service.

4. Data Flow (Numbered Arrows)

Typical online flow:

1. User uploads an image → request goes to Backend API.
2. Backend sends image to Image Processing, then to Deep Learning Model.
3. Model returns prediction → backend prepares response.
4. Result is sent back via API and shown to the user.

Training/admin flow:

1. Admin uploads/curates dataset → stored in Storage and Database.
2. Training job uses dataset to update the deep learning model.
3. Updated model is exposed via API to the application layer for future predictions.