Solution Architecture:smart sorting transfer learning for identifying rotten fruits and vegetables

Date	19 January 2026
Team ID	LTVIP2026TMIDS83701
Project Name	Smart Sorting: Transfer learning for identifying Rotten Fruits and Vegetables
Maximum Marks	4 Marks

1. Introduction

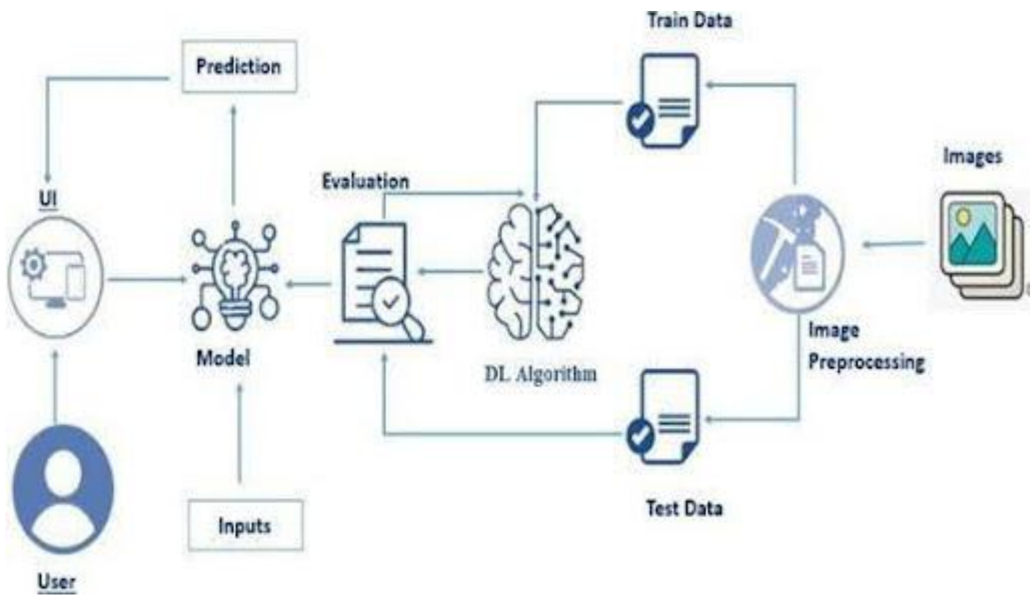
This document outlines the solution architecture for Smart sorting, an AI-powered system designed for the accurate and efficient classification of blood cells. The architecture leverages a combination of deep learning models and a user-friendly web application to provide a robust and scalable solution for pathologists and healthcare professionals.

2. High-Level Architecture Overview

The smart sorting system follows a typical client-server architecture, where a web-based frontend interacts with a Python-based backend that hosts the machine learning model.
The core components include:

- **Client-Side (Web Browser):** User interface for interacting with the system.

- **Web Application Backend (Flask):** Handles user requests, manages image uploads, and orchestrates interactions with the machine learning model.
- **Machine Learning Model (TensorFlow/Keras):** The trained deep learning model responsible for blood cell classification.
- **Storage:** For temporary storage of uploaded images.



3. Detailed Component Breakdown

3.1. Client-Side (Frontend)

- **Technology:** HTML, CSS, JavaScript (standard web technologies).
- **Purpose:** Provides the graphical user interface (GUI) for users to interact with the HematoVision system.
- **Key Functions:**

- **Image Upload:** Allows users to select and upload blood cell images (e.g., `home.html`).
- **Display Results:** Presents the classification prediction and the uploaded image (e.g., `result.html`).
- **User Feedback:** Potentially provides visual cues for upload progress or errors.

3.2. Web Application Backend (Flask)

- **Technology:** Python, Flask framework (`app.py`).
- **Purpose:** Acts as the central hub, receiving requests from the frontend, processing them, and returning responses. It integrates the machine learning model.
- **Key Functions:**
- **API Endpoints:** Defines routes for image upload (`/predict`) and serving web pages (`/`).
- **Image Handling:** Receives uploaded image files, saves them temporarily, and prepares them for model inference.
- **Model Inference Orchestration:** Loads the pre-trained `blood_cell.h5` model and passes the processed image data to it for classification.
- **Result Processing:** Receives the prediction from the model and formats it for display on the frontend.
- **Error Handling:** Manages invalid file types or other processing errors.
- **Templating:** Renders HTML templates (`home.html` , `result.html`) to serve dynamic content to the user.

3.3. Machine Learning Model

- • **Technology:** TensorFlow, Keras, MobileNetV2 / ResNet (fine-tuned model file e.g., fruit_sorting_model.h5).
- • **Purpose:**
  The core intelligence of the system, responsible for accurately classifying fruits and vegetables as **Fresh** or **Rotten**.
- **Key Functions:**
- • **Image Classification:**
  Takes a preprocessed fruit or vegetable image as input and outputs a probability distribution over predefined classes (e.g., Fresh, Rotten).
- • **Feature Extraction:**
  The pre-trained MobileNetV2 (or similar CNN) acts as a powerful feature extractor, identifying patterns such as discoloration, texture changes, bruising, and mold.
- • **Prediction:**
  Provides the final predicted class based on the highest probability score.
- **Training Details:**
- • **Architecture:**
  Pre-trained MobileNetV2 with customized classification layers using transfer learning.
- • **Dataset:**
  Labeled dataset of fresh and rotten fruits and vegetables (e.g., Kaggle dataset).
- • **Training:**
  Trained for multiple epochs using:
- Adam optimizer
- Categorical cross-entropy loss (or binary cross-entropy for two classes)
- • **Accuracy:**
  Achieves high validation accuracy (e.g., ~85–95% depending on dataset quality and tuning).

- • **Model Persistence:**
  The trained model is saved as fruit_sorting_model.h5 for deployment and inference.

- ---

- **3.4 Storage**

- • **Technology:** Local filesystem or cloud storage.

- • **Purpose:**
  Temporarily stores uploaded fruit and vegetable images before processing.

- **Key Functions:**

- • **Temporary Uploads:**
  Uploaded images are stored in a directory such as static/uploads/.

- • **File Management:**
  Images are deleted after processing or after a defined retention period to manage storage space and ensure privacy.

- ---

- **4. Data Flow and Interactions**

- **User Interaction:**
  A user accesses the Smart Sorting web application via a browser, loading the homepage from the backend server.

- **Image Upload:**
  The user selects an image of a fruit or vegetable and uploads it through the web form.
  The HTTP POST request is sent to the /predict endpoint.

- **Backend Processing:**

- The backend application receives the uploaded image.

- The image is temporarily saved in the static/uploads/ directory.

- The image is preprocessed (resizing, normalization) to meet model input requirements.

- The processed image is passed to the loaded fruit_sorting_model.h5 for inference.
- **Model Prediction:**
  The transfer learning model performs classification and returns the predicted label (Fresh or Rotten) along with a confidence score.
- **Result Display:**
- The backend receives the prediction result.
- The result page is rendered with the predicted class and the uploaded image.
- The result is displayed in the user’s browser in real time.
- ---
- **5. Deployment Considerations**
- **• Containerization:**
  The application can be containerized using Docker for consistent deployment.
- **• Cloud Platforms:**
  Suitable for deployment on AWS, Google Cloud, Azure, Render, or Railway.
- **• Scalability:**
  The backend can be horizontally scaled to handle multiple users, and model inference can be offloaded to GPU-enabled cloud services for high-volume processing.
- **• Security:**
  Implement input validation, secure file handling, HTTPS encryption, and secure coding practices.
- ---
- **6. Future Enhancements**
- **• API Integration:**
  Develop a REST API for integration with external warehouse or retail management systems.
- **• Batch Processing:**
  Enable uploading and classification of multiple images simultaneously.

- • **Advanced Confidence Visualization:**
  Display probability graphs or heatmaps indicating spoilage areas.

- • **Database Integration:**
  Store classification results, timestamps, and metadata for auditing and analytics.

- • **User Authentication & Role Management:**
  Implement login, role-based access control (Admin, Inspector, Manager).

- • **Multi-Class Extension:**
  Expand to classify different types of fruits and vegetables along with spoilage severity levels.