**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 25 June 2025 |
| Team ID | LTVIP2025TMID35341 |
| Project Name | Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables |
| Maximum Marks | 4 Marks |

**Technology Stack & Architecture:**

**Smart Sorting** follows a **simple and modular three-tier architecture**, which includes:

### **1. User Interface (Frontend)**

Allows the user to upload fruit or vegetable images through a web page.

### **2. Application Logic (Backend)**

Processes the image, loads the trained deep learning model, and classifies the image.

### **3. Storage & Model Layer**

Holds the trained model (healthy\_vs\_rotten\_best\_model.h5) and manages temporary image storage for prediction.

The application uses a **transfer learning model** (e.g., **VGG16**, **MobileNetV2**) for accurate and fast classification. It is built using **Flask**, a Python micro web framework, for smooth integration of model and web frontend.

**Table-1 : Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | |  | | --- | | Enables image upload through a web page |  |  | | --- | |  | | HTML, CSS, JavaScript |
|  | Application Logic-1 | |  | | --- | |  |  |  | | --- | |  |   Loads and prepares the deep learning model | Python, Flask |
|  | Application Logic-2 | |  | | --- | |  |  |  | | --- | |  |   Performs classification using the transfer learning model | TensorFlow / Keras, VGG16 / MobileNetV2 |
|  | Application Logic-3 | |  | | --- | | Preprocesses image (resizing, normalization) and handles response rendering | | OpenCV, Flask,NumPy |
|  | Database | |  | | --- | | For storing data like past results or logs | | |  | | --- | | SQLite (Optional) | |
|  | Cloud Database | |  | | --- | | If moved to the cloud for scalability |  |  | | --- | |  | | IBM Cloudant / Firebase (Optional) |
|  | File Storage | |  | | --- | | Temporarily stores uploaded images before prediction | | |  | | --- | | Local Filesystem | |
|  | External API-1 | |  |  | | --- | --- | | Could fetch real-time health/environment data |  | |  |  |  | |  |  |  | | OpenWeatherMap API |
|  | External API-2 | Could verify patient identity | UIDAI Aadhaar API |
|  | Machine Learning Model | Classifies the image into 4 blood cell types | MobileNetV2 saved as healthy\_vs\_rotten\_best\_model.h5 |
|  | Infrastructure (Server / Cloud) | Where the app runs | Localhost / Heroku / AWS (optional) |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | |  |  | | --- | --- | | We’ve used popular open tools for better support and cost-efficiency |  | | Flask, TensorFlow, Keras, OpenCV |
|  | Image validation | Verifies allowed image types and handles upload security | Flask utilities, secure path checks |
|  | Scalable Architecture | The modular design lets us move to cloud easily | 3-tier architecture, microservices-ready |
|  | Availability | Can be deployed on platforms with high uptime | Heroku, AWS, Load Balancers (optional) |
|  | Performance | Fast response time using a lightweight model | MobileNetV2, efficient image preprocessing |

**References:**

* [C4 Model for Architecture](https://c4model.com/)
* [IBM Developer Architecture Patterns](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)
* [IBM Cloud Architecture](https://www.ibm.com/cloud/architecture)
* [AWS Architecture Examples](https://aws.amazon.com/architecture)
* [How to Draw Architecture Diagrams](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)