**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 25 june 2025 |
| Team ID | LTVIP2025TMID45798 |
| Project Name | Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

**NutriGaze is a web application that allows users to upload images of fruits and vegetables to classify them as healthy or rotten using a deep learning model. The application uses a VGG16-based model trained with Keras and TensorFlow and is deployed using Flask on a local or cloud server.Example: Order processing during pandemics for offline mode**



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

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | Web interface for image upload and result display | HTML, CSS, JavaScript / Angular Js / React Js etc. |
|  | Application Logic-1 | Handles file upload, model invocation, result rendering | Python(flask) |
|  | Application Logic-2 | Image preprocessing, resizing, normalization | Keras, NumPy, PIL |
|  | Application Logic-3 | Model prediction using pre-trained VGG16 | TensorFlow, Keras |
|  | Database | Stores uploaded files temporarily | Local Filesystem (static/uploads) |
|  | Cloud Database | (Optional if deployed) - Can store metadata, logs, results | Firebase / MongoDB Atlas |
|  | File Storage | Stores uploaded images for prediction | Local filesystem / Cloud bucket |
|  | External API-1 | Not used currently – future use in real-time camera feed | (Future: WebRTC API) |
|  | External API-2 | Not used currently – future use for nutrition info | (Future: Edamam API / FoodData Central) |
|  | Machine Learning Model | Predicts healthy/rotten class using image | VGG16 pre-trained model (custom fine-tuned) |
|  | Infrastructure (Server / Cloud) | App hosting either locally or via Render / Heroku | Flask Dev Server / Render Cloud / AWS EC2. |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Flask, Keras, TensorFlow, NumPy, OpenCV, Matplotlib | All technologies used are open-source |
|  | Security Implementations | File validation, secure filename, restricted file type | Flask security, Werkzeug, SHA filenames |
|  | Scalable Architecture | Modular Flask structure with scope to scale using API microservices | Flask Blueprints (future microservices) |
|  | Availability | Hosted with public access, possible 24/7 availability on Render or Heroku | Render.com / Heroku load-balanced infra |
|  | Performance | Light app optimized with lazy image loading, efficient prediction | TensorFlow backend, batched input handling |

**References:**

**** [**https://keras.io**](https://keras.io)

**** [**https://flask.palletsprojects.com**](https://flask.palletsprojects.com)

**** [**https://developer.ibm.com/patterns/**](https://developer.ibm.com/patterns/)

**** [**https://c4model.com/**](https://c4model.com/)

**** [**https://aws.amazon.com/architecture**](https://aws.amazon.com/architecture)

**** [**https://www.tensorflow.org**](https://www.tensorflow.org)