Project Design Phase-II
Solution Requirements (Functional & Non-functional)

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

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Web Navigation & Information	• Home page describing Smart Sorting purpose and features. • About page showing model details (accuracy, classes, dataset). • Contact page with team and project details.
FR-2	Image Upload & Validation	• Open Predict page and select fruit/vegetable image from device. • Validate file type/size and show error if no image is selected.
FR-3	Image Preprocessing & Inference	• Preprocess uploaded image (resize 224×224, normalize). • Load trained MobileNetV2-based model. • Run model to classify image as Fresh or Rotten.
FR-4	Result Display	• Show predicted label (Fresh/Rotten) on result page. • Display confidence score (%) for prediction. • Show uploaded image preview along with prediction.
FR-5	File Handling & Storage	• Save uploaded image temporarily in uploads/ folder. • Serve image via /uploads/<filename> for display in browser.
FR-6	Model & Dataset Maintenance (Admin)	• Allow developer/admin to update dataset folders (train/test). • Retrain CNN model using train.py and save .h5 file. • Update About-page metrics (accuracy, classes, dataset size).
FR-7	Error Handling	• Show friendly message when prediction fails or file missing. • Log server-side errors for debugging.

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Interface should be clean, responsive, and easy to use on mobile and desktop.
NFR-2	**Security**	Uploaded images are stored in a safe server folder and not publicly browsable by index.
NFR-3	**Reliability**	Model should consistently classify images with ~94% accuracy on test data.
NFR-4	**Performance**	Prediction for one image should complete within a few seconds on server.
NFR-5	**Availability**	Flask app should run reliably during demo/usage with minimal downtime.
NFR-6	**Scalability**	System design should allow moving to a more powerful server or adding GPU if image volume grows.
NFR-7	**Maintainability**	Code should be modular (separate predict.py,train.py, templates, CSS) for easy updates.
NFR-8	**Portability**	Application should run on local machine and be deployable to cloud (e.g., Render/Heroku).