Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	06/02/2026
Team ID	LTVIP2026TMIDS83275
Project Name	Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy
Maximum Marks	4 Marks

Technical Architecture:

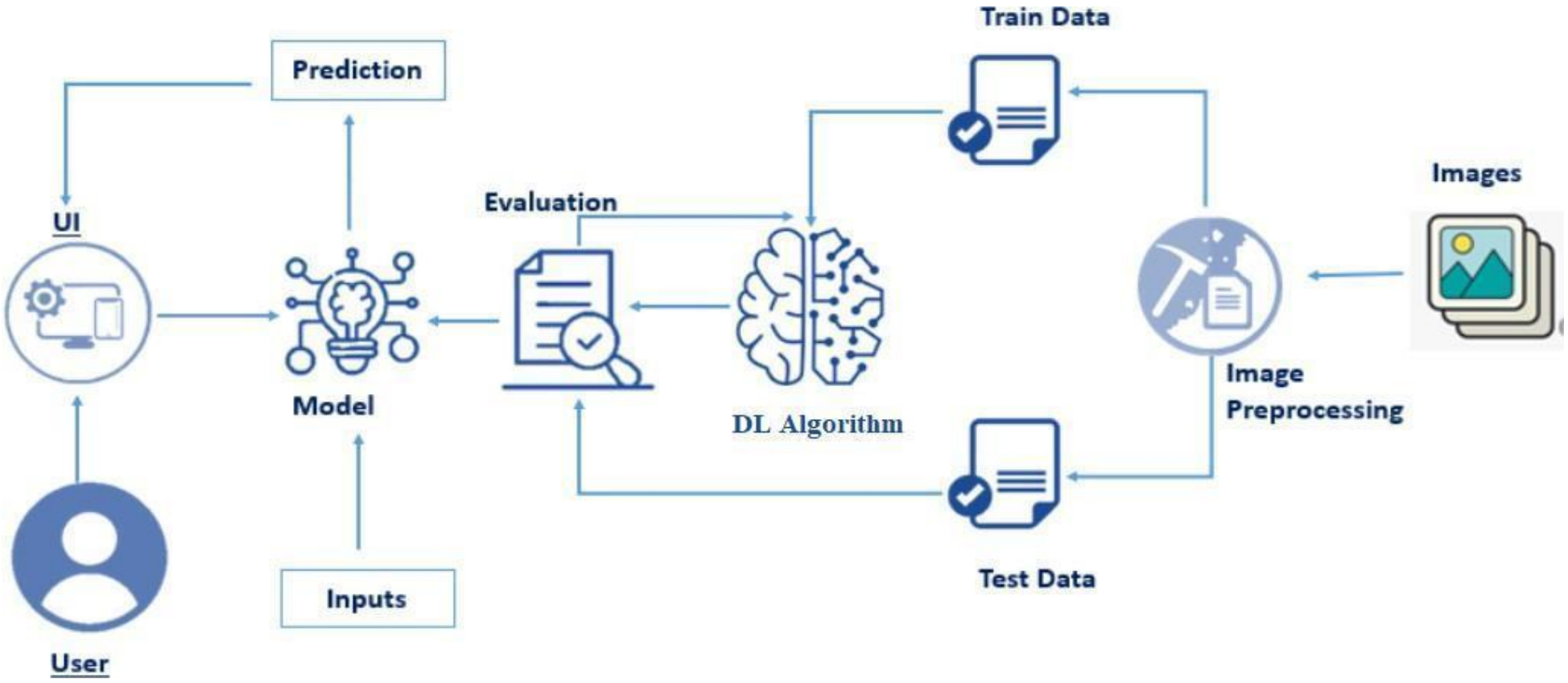

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Web-based interface where users upload retinal fundus images and view prediction results	HTML, CSS, Bootstrap
2.	Application Logic-1	Backend logic for handling image upload, request routing, and response rendering	Python, Flask
3.	Application Logic-2	Image preprocessing logic (resize to 224x224, normalization, array conversion)	Python, NumPy, OpenCV
4.	Application Logic-3	Deep learning model loading and prediction logic	TensorFlow, Keras
5.	Database	No structured database used; prediction results processed in real-time	Not Applicable (Stateless Processing)
6.	Cloud Database	No cloud database integrated; system does not store patient data	Not Applicable
7.	File Storage	Temporary storage of uploaded images during runtime	Local File System (uploads/ folder)
8.	External API-1	No external API currently integrated	Not Applicable
9.	External API-2	Future scope: Integration with hospital management systems or medical APIs	Not Implemented (Future Scope)
10.	Machine Learning Model	Classifies retinal images into diabetic retinopathy severity levels	Convolutional Neural Network (CNN) – TensorFlow/Keras
11.	Infrastructure (Server / Cloud)	Application deployment and hosting environment	Local System (Flask Development Server), Railway Cloud Deployment, Gunicorn

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	The application is built using open-source frameworks for backend development and machine learning model implementation	Python, Flask, TensorFlow, Keras, NumPy
2.	Security Implementations	The system validates input image formats, prevents invalid file uploads, and does not store personal patient data. Basic secure coding practices followed	Flask request validation, File type validation, Python secure handling, HTTPS (Cloud deployment)
3.	Scalable Architecture	The system follows a modular client-server architecture (Presentation Layer – Application Layer – Model Layer). The ML model can be replaced or upgraded independently	Flask (Backend Layer), CNN Model (ML Layer), Railway Cloud Deployment
4.	Availability	The application can be deployed on a cloud platform for 24/7 availability. Cloud hosting ensures remote accessibility	Railway Cloud Platform, Gunicorn WSGI Server
5.	Performance	The system is optimized for real-time inference. Image resizing and normalization reduce computational load. Model inference time < 3 seconds	TensorFlow/Keras optimized model, NumPy preprocessing, CPU-based inference