**Project Design Phase**

**Solution Architecture**

|  |  |
| --- | --- |
| Date | 08/02/2026 |
| Team ID | LTVIP2026TMIDS83275 |
| Project Name | Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy |
| Maximum Marks | 4 Marks |

**Solution Architecture:**

**1. Architecture Overview**

The proposed system follows a Three-Tier Client–Server Architecture consisting of:

1. Presentation Layer (User Interface)
2. Application Layer (Backend Processing)
3. Model Layer (Deep Learning Inference Engine)

The system is designed to automate diabetic retinopathy detection using a trained Convolutional Neural Network (CNN) model integrated with a web-based application.

**2. High-Level Architecture Flow**

User  
↓  
Web Interface (HTML/CSS)  
↓  
Flask Backend Server  
↓  
Image Preprocessing Module  
↓  
Deep Learning CNN Model  
↓  
Prediction Output  
↓  
Result Display on UI

**3. Layer-wise Architecture Description**

**🔹 3.1 Presentation Layer (Front-End)**

**Purpose:**Handles user interaction.

**Components:**

* Web interface for image upload
* Result display section
* Severity classification output

**Technologies Used:**

* HTML
* CSS
* JavaScript (optional)
* Bootstrap (optional)

**Function:**

* Accept retinal fundus image (.jpg/.png)
* Send image to backend for processing
* Display prediction result

**🔹 3.2 Application Layer (Backend Logic)**

**Purpose:**Controls business logic and communication between UI and model.

**Components:**

* Flask Server
* Image Validation
* Image Preprocessing Module
* Model Loader

**Technologies Used:**

* Python
* Flask
* NumPy
* OpenCV / PIL

**Functions:**

* Receives uploaded image
* Resizes image (e.g., 224x224)
* Normalizes pixel values
* Converts image to tensor format
* Sends processed image to CNN model
* Receives prediction
* Returns result to UI

**🔹 3.3 Model Layer (Deep Learning Engine)**

**Purpose:**Performs classification of diabetic retinopathy severity.

**Model Type:**

* Convolutional Neural Network (CNN)

**Technologies Used:**

* TensorFlow
* Keras

**Model Operations:**

* Feature extraction via convolution layers
* Pooling layers for dimensionality reduction
* Fully connected layers
* Softmax output layer for classification

**Output Classes Example:**

* No DR
* Mild
* Moderate
* Severe
* Proliferative DR

**4. Data Flow Description**

1. User uploads fundus image.
2. Image stored temporarily on server.
3. Image preprocessing applied.
4. Processed image fed to trained CNN model.
5. Model predicts severity class.
6. Result displayed to user.

**5. Deployment Architecture**

**🔹 Local Deployment**

* Python Virtual Environment
* Flask Development Server
* Local CPU execution

**🔹 Cloud Deployment (Optional)**

* Cloud Platform (Railway / AWS / Azure)
* Gunicorn WSGI Server
* Containerized Deployment (Docker – optional)

**6. Architecture Characteristics**

* Modular design
* Model separated from UI
* Easy retraining capability
* Stateless inference
* Scalable cloud-compatible structure

**7. Security Considerations**

* Secure file upload validation
* Limited file type acceptance
* No permanent storage of patient images
* HTTPS-based deployment (for cloud)

**Solution Architecture Diagram**

**Figure 1: Architecture and Data Flow Diagram**

**User**

**↓**

**Web Interface (Upload Image)**

**↓**

**Flask Backend**

**↓**

**Image Preprocessing**

**↓**

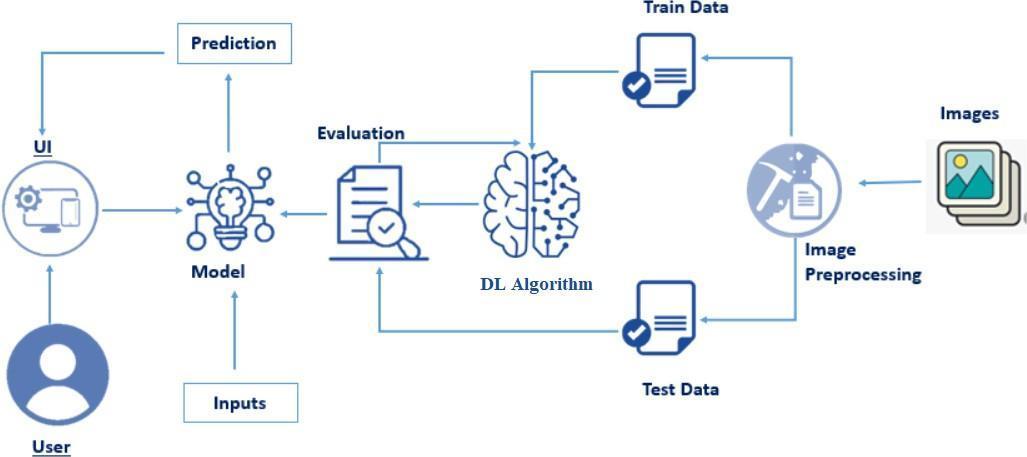
**CNN Model (.h5)**

**↓**

**Prediction Output**

**↓**

**Result Display on UI**



**Flow:**

Fundus Image Dataset → Preprocessing → Train/Test Split → CNN Model Training → Evaluation → Model Saving (.h5) → Web Integration → Prediction → User

**Training Phase Flow:**

Dataset → Preprocessing → Train/Test Split → CNN Training → Evaluation → Save Model