Full Stack Development with MERN

Project Documentation format

1. Introduction

• **Project Title:** Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy.
• **Team Members:** Madineni Bhavana , Kunte Upendranath Rao, Poojari Niteesh Kumar, Muchumarri Suresh

2. Project Overview

Purpose: To provide an automated, AI-driven screening tool that detects various stages of Diabetic Retinopathy (DR) from fundus images to assist clinicians in early diagnosis.

Features: Secure user authentication, high-resolution fundus image upload, real-time AI inference, and historical report tracking.

3. Architecture

Frontend: Built with React, utilizing functional components and hooks for state management and Axios for API communication.
Backend: Developed using Node.js and Express.js, acting as a middleware to handle user requests and interface with the Python/Deep Learning model
Database: MongoDB stores user profiles, encrypted passwords, and metadata for uploaded images and diagnosis results.

4. Setup Instructions

• **Prerequisites:** Node.js (v16+), MongoDB Atlas or local instance, and Python 3.8+ (for the DL model).

Installation:

- git clone [repository-url]
- cd client && npm install
- cd server && npm install
- Create a .env file with MONGO_URI and JWT_SECRET.

5. Folder Structure

• **Client**: Includes /src/components (UI elements), /src/pages (Login, Dashboard, Analysis), and /src/services (API calls).

• **Server**: Includes /routes (API endpoints), /models (Mongoose schemas), and /controllers (Logic for uploads and AI triggering).

6. Running the Application

• Provide commands to start the frontend and backend servers locally. ○
Frontend: Run npm start in the client directory to launch the React app..
Backend: Run npm start in the server directory to launch the Express server.

7. API Documentation

 POST /api/auth/register: Registers a new user/clinician.

 POST /api/upload: Accepts a fundus image (multipart/form-data)

and returns the DR stage classification.

 GET /api/reports: Retrieves previous diagnosis history for the

logged-in user.

8. Authentication

 Method: JSON Web Tokens (JWT) are used for

secure authentication.

 Process: Upon login, the server issues a token stored

in the browser's local storage; this token is required for all

authorized API requests.

9. User Interface

Dashboard: A clean interface showing upload progress and result visualizations

10. Testing

 Strategy: Functional and performance testing using manual checks and automated scripts.

 Tools: Postman for API testing, Jest for unit testing, and Chrome DevTools for UI performance.

12. Known Issues

High-resolution images (>10MB) may experience slight

latency during upload.

13. Future Enhancements

Integration of Explainable AI (XAI) to highlight specific lesions

(hemorrhages/exudates) on the fundus image.