Project Documentation Format

1. Introduction

Project Title: *HematoVision: Advanced Blood Cell Classification Using Transfer Learning*

Team Members:

Team lead: Baddula Krishnapriya – Frontend Developer

Team member: Anil Kumar –Backend Developer

Team member: Aineru Monish – Data Scientist & Model Trainer

Team member: Amburu Harshith Kumar – Deployment & Integration

2. Project Overview

Purpose: To develop a web-based platform that uses deep learning to classify blood cell types—Eosinophil, Lymphocyte, Monocyte, and Neutrophil—from microscopic images, aiding medical diagnostics.

Features:

- User-friendly interface for uploading blood cell images
- AI-based classification using transfer learning
- Visualization of model predictions and confidence scores
- Admin dashboard for dataset management and monitoring

3. Architecture

Frontend: Built using React.js with Axios for API calls. Component-based structure with routing managed via React Router. UI designed using Material UI.

Backend: Node.js with Express.js serves the API routes for prediction, authentication, and image handling. Python-based AI model is triggered from the backend using child processes or Flask microservice.

Database: MongoDB stores user data, classification results, image metadata, and logs. Mongoose is used for schema definition and queries.

4. Setup Instructions

Prerequisites:

- Node.js (v16+)
- MongoDB (local or cloud instance)
- Python (for AI model)
- Git

Installation:

```
git clone https://github.com/your-repo/HematoVision.git

cd HematoVision

npm install

cd client

npm install
```

```
cd ..

touch .env  # and add your environment variables
```

5. Folder Structure

Client/

```
src/

|-- components/

|-- pages/

|-- services/

|-- App.js

|-- index.js
```

Server/

```
routes/

|-- auth.js

|-- predict.js

controllers/

models/

app.js
```

6. Running the Application

Frontend:

bash

```
cd client
npm start
```

Backend:

bash

```
npm start
```

7. API Documentation

- **POST /api/predict** Accepts an image file and returns the expected class with a confidence score.
- **POST /api/auth/login** Authenticates the admin.
- **GET /api/history** Returns user classification history.

8. Authentication

Implemented JWT-based authentication.

- Token stored in HTTP-only cookies
- Middleware verifies and protects secure routes

9. User Interface

Includes:

- Upload form for image input
- Result display with predicted label and confidence
- Admin dashboard for user/image stats

10. Testing

- **Frontend**: Jest + React Testing Library
- **Backend**: Mocha + Chai
- **Model**: Accuracy and F1-score evaluation on test data using sklearn

11. Screenshots or Demo:

DEMO:

https://drive.google.com/file/d/1MsCRDd6AtZKYjByMoKI8oh_3gh-hCZ5R/view?usp=drive_link

Image Link:

https://bing.com/th/id/BCO.d96121a9-ca64-4b62-999c-0673c259ec85.png

12. Known Issues:

- Initial load may be delayed due to model warm-up
- Mobile responsiveness is limited for the dashboard

13. Future Enhancements:

- Integrate a feedback loop for model retraining
- Deploy on cloud with GPU support (e.g., Azure ML, AWS EC2)
- Add support for additional cell types