1. Introduction

Project Title:
HematoVision – Blood Cell Classification System Using Transfer Learning

Team Members:

- Member 1 – Frontend Development (React)
- Member 2 – Backend Development (Node.js / Express)
- Member 3 – Database Management (MongoDB)
- Member 4 – Model Integration / Testing

---

2. Project Overview

Purpose:
HematoVision is a full stack web application designed to classify blood cell images using a Transfer Learning-based deep learning model. The system enables users to upload images and receive predictions along with confidence scores.

Features:

- Blood cell image upload
- Real-time prediction display
- Confidence score visualization
- Error handling for invalid inputs
- Responsive user interface
- Model integration with backend

---

3. Architecture

Frontend (React)

The frontend is developed using **React.js**, providing a responsive and interactive user interface.

Responsibilities:

- Image upload interface
- Display prediction results
- Error notifications

- Loading indicators

Backend (Node.js & Express.js)

The backend is implemented using **Node.js** and **Express.js**, acting as the communication layer between the UI, database, and ML services.

Responsibilities:

- API endpoints
- Request validation
- Model communication
- Response handling

Database (MongoDB)

MongoDB is used for storing application-related data.

Stored Data (Example):

- User details (if authentication used)
- Prediction logs
- Uploaded image metadata

---

4. Setup Instructions

Prerequisites

Ensure the following software is installed:

- Node.js
- MongoDB
- npm (Node Package Manager)
- Git

Installation

Step 1 – Clone Repository

```
git clone <https://github.com/Sweekruti28/HEMATOVISION-ADVANCED-BLOOD-CELL-
CLASSIFICATION-USING-TRANSFER-LEARNING>
cd hematovision
```

Step 2 – Install Dependencies

Client (Frontend):

```
cd client
npm install
```

Server (Backend):

```
cd server
npm install
```

Step 3 – Environment Variables

Create `.env` file in server folder:

```
PORT=5000
MONGO_URI=your_mongodb_connection_string
```

---

5. Folder Structure

Client (React Frontend)

```
client/
 ├── public/
 ├── src/
 │    ├── components/
 │    ├── pages/
 │    ├── services/
 │    ├── App.js
 │    └── index.js
 │
```

Server (Node.js Backend)

```
server/
```

```
├── routes/
├── controllers/
├── models/
├── middleware/
├── server.js
└── config/
```

---

6. Running the Application

Frontend

```
cd client
npm start
```

Backend

```
cd server
npm start
```

---

7. API Documentation

1. Upload Image

Endpoint:
`POST /api/predict`

Description:
Accepts blood cell image and returns classification result.

Request:

- Method: POST
- Content-Type: multipart/form-data

Response Example:

```
{
  "predicted_class": "Neutrophil",
  "confidence_score": 0.94
}
```

---

2. Prediction Logs (Optional)

Endpoint:
`GET /api/logs`

Description:
Fetches stored prediction history.

---

8. Authentication

Authentication is handled using **JWT (JSON Web Tokens)**.

Mechanism:

- User login → Token generation
- Token validation via middleware
- Protected API routes

---

9. User Interface

The application provides:

- Image Upload Page
- Prediction Results Display
- Confidence Score Visualization
- Error Handling Messages

---

10. Testing

Testing Strategy:

- Functional Testing
- API Testing
- UI Testing
- Performance Testing

Tools Used:

- Postman (API Testing)
- Browser Testing
- Manual Validation

---

11. Screenshots / Demo

Include:

- UI Screenshots
- Prediction Output
- System Workflow

---

12. Known Issues

- Model performance depends on dataset quality
- Slight latency during prediction requests
- Requires stable backend connection

---

13. Future Enhancements

- Real-time camera image capture
- Advanced visualization dashboard
- Multi-class disease detection
- Cloud deployment
- Improved inference speed