# Artificial Intelligence and Machine Learning

## Project Documentation

### 1. Introduction

• Project Title: Hematovision: Advanced Blood Cell Classification Using Transfer Learning

• Team ID: LTVIP2025TMID42022

• Team Members:

- P. Mahitha – Frontend Developer

- M. Shivateja – Backend Developer

- S. Sindhu – AI/ML Specialist

- B. Akhila – Database & Integration

### 2. Project Overview

• Purpose: Hematovision aims to automate the classification of blood cells using AI-based transfer learning models. This solution helps reduce diagnosis time, assists lab technicians, and increases accuracy in detecting abnormal cells.

• Features:

- Secure image upload

- AI-powered classification (RBC, WBC, Platelets, Abnormal)

- PDF report generation

- Role-based login (Technician, Doctor, Admin)

- Web-based dashboard interface

### 3. Architecture

• Frontend: Built using React.js with Tailwind CSS for responsive design. Axios is used for API calls and protected routes are implemented using React Router.

• Backend: Node.js with Express.js handles routing, API logic, and model integration. RESTful APIs facilitate communication between the frontend and backend.

• Database: MongoDB stores user credentials, uploaded file metadata, and classification logs. GridFS handles file storage.

### 4. Setup Instructions

• Prerequisites: Node.js, MongoDB, Git, Python (for ML model service)

• Installation:

git clone https://github.com/your-repo/hematovision

cd hematovision

npm install

cd client

npm install

• Environment Setup: Add .env files with MongoDB URI, Port, and JWT Secret

### 5. Folder Structure

• Client (React): /components, /pages, /services

• Server (Node.js): /routes, /controllers, /models, /ml

### 6. Running the Application

Frontend:  
cd client  
npm start

Backend:  
cd server  
npm start

### 7. API Documentation

• /api/upload – POST – Uploads blood cell image

• /api/classify – GET – Returns classification result

• /api/report – GET – Downloads diagnostic report

• /api/login – POST – Authenticates user

### 8. Authentication

• JWT is used for user authentication.  
• Passwords are hashed using bcrypt.  
• Protected routes and roles (Admin, Doctor, Technician) managed via middleware.

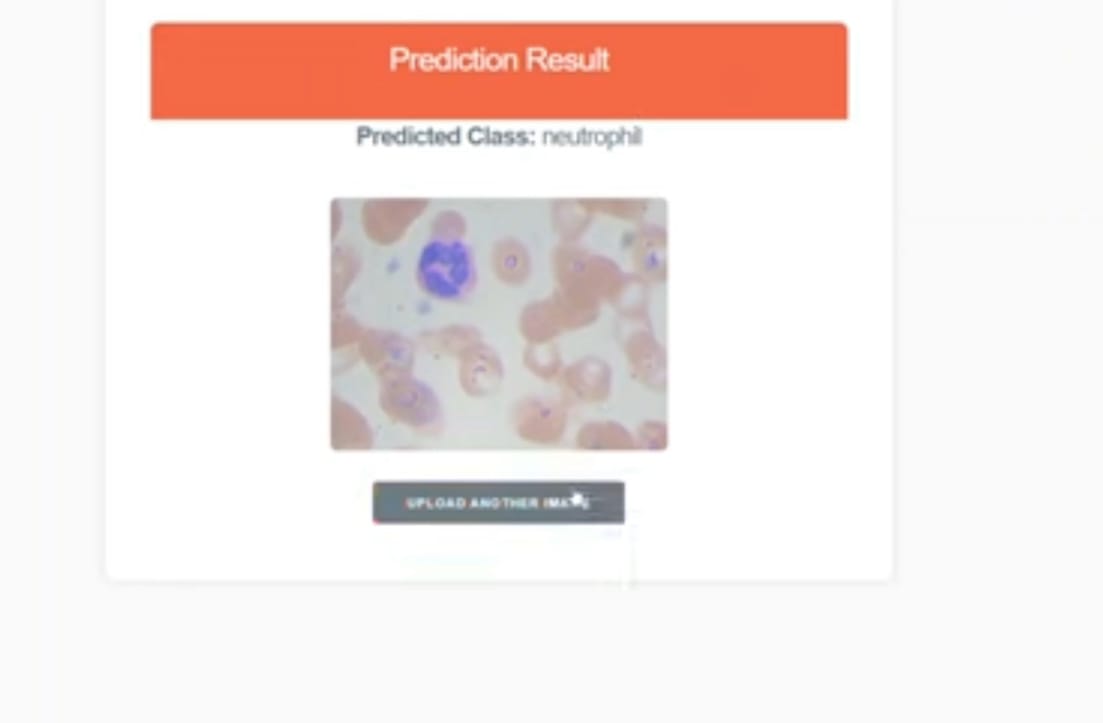
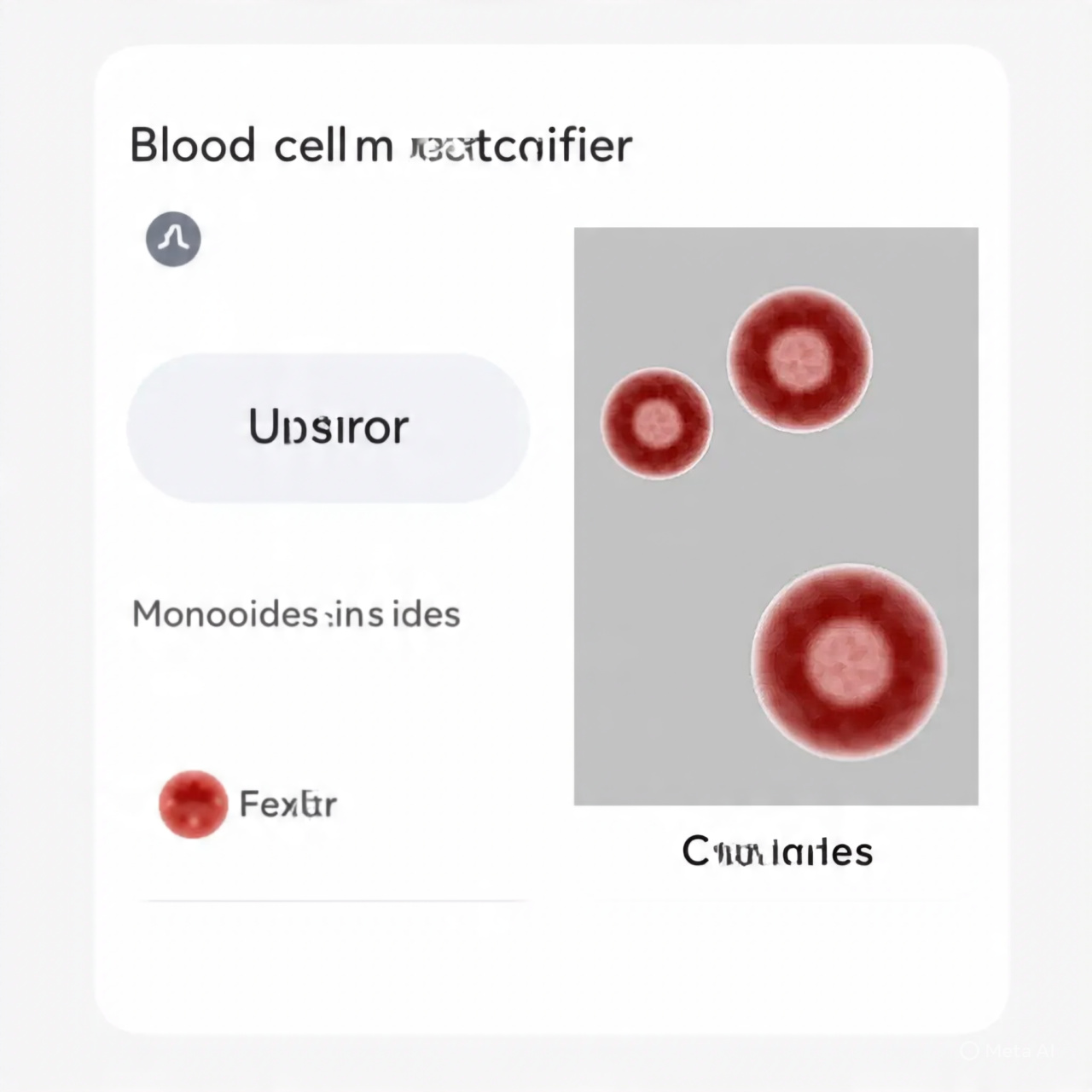
### 9. User Interface

• Web dashboard with role-based access  
• Upload form  
• Classification results view  
• PDF download option

### 10. Testing

• Unit testing with Jest  
• API testing using Postman  
• Manual test cases for frontend workflows

### 11. Screenshots or Demo



### 12. Known Issues

• High-resolution images might cause slight latency in classification  
• Limited to pre-trained model categories (may miss rare cell types)

### 13. Future Enhancements

• Expand classification categories (e.g., sickle cells, leukemia indicators)  
• Mobile app version  
• Multilingual UI for accessibility  
• Integration with EHR systems