File: .gitignore

```
# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*
lerna-debug.log*

node_modules
dist
dist-ssr
*.local

# Editor directories and files
.vscode/*
!.vscode/extensions.json
.idea
.DS_Store
*.suo
*.ntvs*
*.njsproj
*.sln
*.sw?
```

File: App.tsx

```
import React, { useState } from 'react';
import Layout from './components/Layout';
import Dashboard from './components/Dashboard';
import AnalysisView from './components/AnalysisView';
import EducationView from './components/EducationView';
import ModelEvaluation from './components/ModelEvaluation';
import PresentationView from './components/PresentationView';
import HelpView from './components/HelpView';
import { ViewState } from './types';

const App: React.FC = () => {
  const [currentView, setCurrentView] = useState<ViewState>(ViewState.DASHBOARD);

  const renderContent = () => {
    switch (currentView) {
      case ViewState.DASHBOARD:
        return <Dashboard onChangeView={setCurrentView} />;
      case ViewState.ANALYSIS:
        return <AnalysisView />;
      case ViewState.MODEL_EVALUATION:
        return <ModelEvaluation />;
      case ViewState.EDUCATION:
        return <EducationView />;
      case ViewState.PRESENTATION:
        return <PresentationView />;
      case ViewState.HELP:
        return <HelpView />;
      default:
        return <Dashboard onChangeView={setCurrentView} />;
    }
  };

  return (
    <Layout currentView={currentView} onViewChange={setCurrentView}>
      {renderContent()}
    </Layout>
  );
};

export default App;
```

File: README.md

```
<div align="center">
<img width="1200" height="475" alt="GHBanner" src="https://github.com/user-attachments/assets/0aa67016-6eaf-
</div>

# Run and deploy your AI Studio app

This contains everything you need to run your app locally.

View your app in AI Studio: https://ai.studio/apps/drive/1lnOl1am1B6W1MQvRy4CGbn81J6msupiu

## Run Locally

**Prerequisites:**  Node.js


1. Install dependencies:
   `npm install`
2. Set the `GEMINI_API_KEY` in [.env.local](.env.local) to your Gemini API key
3. Run the app:
   `npm run dev`
```

File: constants.ts

```
import { CellType } from "./types";

export const APP_NAME = "HematoVision";
export const APP_VERSION = "1.0.0";

export const CELL_TYPES = [
  CellType.EOSINOPHIL,
  CellType.LYMPHOCYTE,
  CellType.MONOCYTE,
  CellType.NEUTROPHIL
];

export const SYSTEM_INSTRUCTION = `
You are HematoVision, an advanced AI pathologist assistant specializing in hematology and blood cell morphol
Your task is to analyze microscopic images of blood cells.
Classify the dominant cell in the image into one of these categories: Eosinophil, Lymphocyte, Monocyte, Neut
If the image is unclear or does not contain a blood cell, classify as Unknown.

Provide a response in JSON format with the following structure:
{
  "cellType": "Eosinophil" | "Lymphocyte" | "Monocyte" | "Neutrophil" | "Unknown",
  "confidence": number (0-100),
  "description": "A concise description of the visual features observed.",
  "morphology": ["List", "of", "key", "morphological", "features"],
  "clinicalSignificance": "Brief note on what elevated levels of this cell might indicate (e.g., bacterial i
}
`;

export const EDUCATIONAL_PROMPT = `
Generate a challenging multiple-choice quiz question for a medical student about blood cell morphology.
The question should focus on distinguishing features between Eosinophils, Lymphocytes, Monocytes, and Neutro
Provide the response in JSON format:
{
  "question": "The question text",
  "options": ["Option A", "Option B", "Option C", "Option D"],
  "correctAnswer": "The correct option string",
  "explanation": "Detailed explanation of why the answer is correct."
}
`;
```

File: index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>HematoVision</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap" rel="s
    <style>
      body {
        font-family: 'Inter', sans-serif;
      }
      /* Custom scrollbar for webkit */
      ::-webkit-scrollbar {
        width: 8px;
      }
      ::-webkit-scrollbar-track {
        background: #f1f1f1;
      }
      ::-webkit-scrollbar-thumb {
        background: #cbd5e1;
        border-radius: 4px;
      }
      ::-webkit-scrollbar-thumb:hover {
        background: #94a3b8;
      }
    </style>
  <script type="importmap">
{
  "imports": {
    "@google/genai": "https://esm.sh/@google/genai@^1.41.0",
    "react-dom/": "https://esm.sh/react-dom@^19.2.4/",
    "lucide-react": "https://esm.sh/lucide-react@^0.574.0",
    "recharts": "https://esm.sh/recharts@^3.7.0",
    "react/": "https://esm.sh/react@^19.2.4/",
    "react": "https://esm.sh/react@^19.2.4"
  }
}
</script>
<link rel="stylesheet" href="/index.css">
</head>
  <body class="bg-slate-50 text-slate-900 antialiased">
    <div id="root"></div>
  <script type="module" src="/index.tsx"></script>
</body>
</html>
```

File: index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

const rootElement = document.getElementById('root');
if (!rootElement) {
  throw new Error("Could not find root element to mount to");
}

const root = ReactDOM.createRoot(rootElement);
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

File: metadata.json


```
{
  "name": "HematoVision",
  "description": "Advanced Blood Cell Classification using AI. Analyzes blood smear images to identify Eosin
  "requestFramePermissions": [
    "camera"
  ]
}
```

File: package.json

```
{
  "name": "hematovision",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "@google/genai": "^1.41.0",
    "react-dom": "^19.2.4",
    "lucide-react": "^0.574.0",
    "recharts": "^3.7.0",
    "react": "^19.2.4"
  },
  "devDependencies": {
    "@types/node": "^22.14.0",
    "@vitejs/plugin-react": "^5.0.0",
    "typescript": "~5.8.2",
    "vite": "^6.2.0"
  }
}
```

File: tsconfig.json

```
{
  "compilerOptions": {
    "target": "ES2022",
    "experimentalDecorators": true,
    "useDefineForClassFields": false,
    "module": "ESNext",
    "lib": [
      "ES2022",
      "DOM",
      "DOM.Iterable"
    ],
    "skipLibCheck": true,
    "types": [
      "node"
    ],
    "moduleResolution": "bundler",
    "isolatedModules": true,
    "moduleDetection": "force",
    "allowJs": true,
    "jsx": "react-jsx",
    "paths": {
      "@/*": [
        "./*"
      ]
    },
    "allowImportingTsExtensions": true,
    "noEmit": true
  }
}
```

File: types.ts

```
export enum ViewState {
  DASHBOARD = 'DASHBOARD',
  ANALYSIS = 'ANALYSIS',
  EDUCATION = 'EDUCATION',
  MODEL_EVALUATION = 'MODEL_EVALUATION',
  PRESENTATION = 'PRESENTATION',
  HELP = 'HELP',
  SETTINGS = 'SETTINGS'
}

export enum CellType {
  EOSINOPHIL = 'Eosinophil',
  LYMPHOCYTE = 'Lymphocyte',
  MONOCYTE = 'Monocyte',
  NEUTROPHIL = 'Neutrophil',
  UNKNOWN = 'Unknown'
}

export interface AnalysisResult {
  cellType: CellType;
  confidence: number;
  description: string;
  morphology: string[];
  clinicalSignificance?: string;
}

export interface QuizQuestion {
  imageUrl: string;
  correctType: CellType;
  options: CellType[];
  explanation: string;
}
```

File: vite.config.ts

```
import path from 'path';
import { defineConfig, loadEnv } from 'vite';
import react from '@vitejs/plugin-react';

export default defineConfig(({ mode }) => {
    const env = loadEnv(mode, '.', '');
    return {
      server: {
        port: 3000,
        host: '0.0.0.0',
      },
      plugins: [react()],
      define: {
        'process.env.API_KEY': JSON.stringify(env.GEMINI_API_KEY),
        'process.env.GEMINI_API_KEY': JSON.stringify(env.GEMINI_API_KEY)
      },
      resolve: {
        alias: {
          '@': path.resolve(__dirname, '.'),
        }
      }
    };
});
```

File: services/geminiService.ts

```
import { GoogleGenAI, Type } from "@google/genai";
import { SYSTEM_INSTRUCTION } from "../constants";
import { AnalysisResult, CellType } from "../types";

// Initialize Gemini Client
// IMPORTANT: API_KEY is injected by the environment.
const ai = new GoogleGenAI({ apiKey: process.env.API_KEY });

export const analyzeBloodCellImage = async (base64Image: string): Promise<AnalysisResult> => {
  try {
    const response = await ai.models.generateContent({
      model: 'gemini-3-pro-preview',
      contents: {
        parts: [
          {
            inlineData: {
              mimeType: 'image/png', // Assuming PNG for simplicity, Gemini handles JPEGs too
              data: base64Image
            }
          },
          {
            text: "Analyze this blood cell image. Identify the cell type, confidence, morphology, and signif
          }
        ]
      },
      config: {
        systemInstruction: SYSTEM_INSTRUCTION,
        responseMimeType: "application/json",
        responseSchema: {
          type: Type.OBJECT,
          properties: {
            cellType: { type: Type.STRING, enum: [CellType.EOSINOPHIL, CellType.LYMPHOCYTE, CellType.MONOCYT
            confidence: { type: Type.NUMBER },
            description: { type: Type.STRING },
            morphology: { type: Type.ARRAY, items: { type: Type.STRING } },
            clinicalSignificance: { type: Type.STRING }
          },
          required: ["cellType", "confidence", "description", "morphology"]
        }
      }
    });

    const text = response.text;
    if (!text) throw new Error("No response from AI");

    return JSON.parse(text) as AnalysisResult;
  } catch (error) {
    console.error("Analysis failed:", error);
    throw new Error("Failed to analyze image. Please ensure the image is clear.");
  }
};

export const generateQuiz = async (): Promise<any> => {
  try {
    const response = await ai.models.generateContent({
      model: 'gemini-3-flash-preview',
      contents: "Generate a hematology quiz question.",
      config: {
        systemInstruction: "You are a medical educator. Create a quiz question about blood cells.",
        responseMimeType: "application/json",
        responseSchema: {
            type: Type.OBJECT,
            properties: {
                question: { type: Type.STRING },
```

```
                options: { type: Type.ARRAY, items: { type: Type.STRING }},
                correctAnswer: { type: Type.STRING },
                explanation: { type: Type.STRING }
            }
        }
      }
    });
    const text = response.text;
    if (!text) throw new Error("No response from AI");
    return JSON.parse(text);
  } catch (error) {
    console.error("Quiz generation failed", error);
    throw error;
  }
}
```

File: services/hospitalService.ts

```
import { GoogleGenAI, Type } from "@google/genai";
import { SYSTEM_INSTRUCTION } from "../constants";
import { AnalysisResult, CellType } from "../types";

// Initialize AI Client
// IMPORTANT: API_KEY is injected by the environment.
const ai = new GoogleGenAI({ apiKey: process.env.API_KEY });

export const analyzeBloodCellImage = async (base64Image: string): Promise<AnalysisResult> => {
  try {
    const response = await ai.models.generateContent({
      model: 'gemini-3-pro-preview',
      contents: {
        parts: [
          {
            inlineData: {
              mimeType: 'image/png', // Assuming PNG for simplicity, Gemini handles JPEGs too
              data: base64Image
            }
          },
          {
            text: "Analyze this blood cell image. Identify the cell type, confidence, morphology, and signif
          }
        ]
      },
      config: {
        systemInstruction: SYSTEM_INSTRUCTION,
        responseMimeType: "application/json",
        responseSchema: {
          type: Type.OBJECT,
          properties: {
            cellType: { type: Type.STRING, enum: [CellType.EOSINOPHIL, CellType.LYMPHOCYTE, CellType.MONOCYT
            confidence: { type: Type.NUMBER },
            description: { type: Type.STRING },
            morphology: { type: Type.ARRAY, items: { type: Type.STRING } },
            clinicalSignificance: { type: Type.STRING }
          },
          required: ["cellType", "confidence", "description", "morphology"]
        }
      }
    });

    const text = response.text;
    if (!text) throw new Error("No response from AI");

    return JSON.parse(text) as AnalysisResult;
  } catch (error) {
    console.error("Analysis failed:", error);
    throw new Error("Failed to analyze image. Please ensure the image is clear.");
  }
};

export const generateQuiz = async (): Promise<any> => {
  try {
    const response = await ai.models.generateContent({
      model: 'gemini-3-flash-preview',
      contents: "Generate a hematology quiz question.",
      config: {
        systemInstruction: "You are a medical educator. Create a quiz question about blood cells.",
        responseMimeType: "application/json",
        responseSchema: {
            type: Type.OBJECT,
            properties: {
                question: { type: Type.STRING },
```

```
                options: { type: Type.ARRAY, items: { type: Type.STRING }},
                correctAnswer: { type: Type.STRING },
                explanation: { type: Type.STRING }
            }
        }
      }
    });
    const text = response.text;
    if (!text) throw new Error("No response from AI");
    return JSON.parse(text);
  } catch (error) {
    console.error("Quiz generation failed", error);
    throw error;
  }
}
```

File: components/AnalysisView.tsx

```
import React, { useState, useRef } from 'react';
import CameraCapture from './CameraCapture';
import { analyzeBloodCellImage } from '../services/hospitalService';
import { AnalysisResult } from '../types';
import { Loader2, Camera } from 'lucide-react';

const AnalysisView: React.FC = () => {
  const [selectedFile, setSelectedFile] = useState<File | null>(null);
  const [previewUrl, setPreviewUrl] = useState<string | null>(null);
  const [isAnalyzing, setIsAnalyzing] = useState(false);
  const [result, setResult] = useState<AnalysisResult | null>(null);
  const [error, setError] = useState<string | null>(null);
  const [showCamera, setShowCamera] = useState(false);
  const fileInputRef = useRef<HTMLInputElement>(null);

  const handleFileChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    if (event.target.files && event.target.files[0]) {
      const file = event.target.files[0];
      setSelectedFile(file);
      const url = URL.createObjectURL(file);
      setPreviewUrl(url);
      setResult(null);
      setError(null);
      setShowCamera(false);
    }
  };

  const handleCameraCapture = (file: File) => {
    setSelectedFile(file);
    const url = URL.createObjectURL(file);
    setPreviewUrl(url);
    setResult(null);
    setError(null);
    setShowCamera(false);
  };

  const handleAnalyze = async () => {
    if (!selectedFile) return;

    setIsAnalyzing(true);
    setError(null);

    try {
      const reader = new FileReader();
      reader.onloadend = async () => {
        const base64String = reader.result as string;
        // Handle both png and jpeg prefixes
        const base64Data = base64String.includes('base64,')
            ? base64String.split('base64,')[1]
            : base64String;

        try {
          const analysisResult = await analyzeBloodCellImage(base64Data);
          setResult(analysisResult);
        } catch (err: any) {
          setError(err.message || "Failed to analyze image");
        } finally {
          setIsAnalyzing(false);
        }
      };
      reader.readAsDataURL(selectedFile);
    } catch (err) {
      setError("Error processing image file");
      setIsAnalyzing(false);
```

```
  }
};

const resetAnalysis = () => {
  setSelectedFile(null);
  setPreviewUrl(null);
  setResult(null);
  setError(null);
  if (fileInputRef.current) {
    fileInputRef.current.value = '';
  }
};

// Result View
if (result) {
  return (
    <div className="flex flex-col items-center justify-center min-h-[80vh] p-4 bg-slate-50">
      <div className="bg-white shadow-xl rounded-lg overflow-hidden w-full max-w-2xl border border-slate-2
        <div className="bg-[#e74c3c] p-4 py-6 text-center">
          <h2 className="text-3xl font-bold text-white">Prediction Result</h2>
        </div>

        <div className="p-8 flex flex-col items-center">
          <h3 className="text-xl text-slate-700 font-bold mb-6">
            Predicted Class: <span className="text-slate-900 font-normal">{result.cellType.toLowerCase()}<
          </h3>

          {previewUrl && (
            <div className="mb-8 p-1 bg-white border border-slate-200 shadow-sm rounded-lg">
              <img
                src={previewUrl}
                alt="Analyzed blood cell"
                className="max-h-64 object-contain rounded"
              />
            </div>
          )}

          {/* Additional info hidden in screenshot but kept for utility */}
          <div className="w-full bg-slate-50 p-4 rounded-lg mb-6 border border-slate-100 text-left text-sm
             <p className="mb-1"><span className="font-semibold text-slate-700">Confidence:</span> {result
             <p><span className="font-semibold text-slate-700">Morphology:</span> {result.morphology.join(
          </div>

          <button
            onClick={resetAnalysis}
            className="bg-[#e74c3c] hover:bg-[#c0392b] text-white font-semibold py-2 px-6 rounded shadow t
          >
            Upload Another Image
          </button>
        </div>
      </div>
    </div>
  );
}

// Input View
return (
  <div className="flex flex-col items-center min-h-[80vh] p-4 bg-slate-50">
    <div className="bg-white shadow-xl rounded-lg overflow-hidden w-full max-w-5xl border border-slate-200

      {/* Header */}
      <div className="bg-[#e74c3c] p-4 py-6 text-center">
        <h1 className="text-3xl md:text-4xl font-bold text-white">Welcome to the HematoVision</h1>
      </div>

      <div className="p-8 md:p-12 space-y-10">

        {/* About Section */}
```

```
<section>
  <h2 className="text-2xl font-bold text-[#e74c3c] mb-3">About Blood Cells</h2>
  <p className="text-slate-600 leading-relaxed">
    Blood cells are vital components of our body, playing essential roles in immunity, oxygen tran
  </p>
</section>

{/* Predict Section */}
<section className="pt-4">
  <h2 className="text-2xl font-bold text-[#e74c3c] mb-3 text-center">Predict Blood Cell Type</h2>
  <p className="text-slate-600 mb-8 text-center">
    Upload an image of a blood cell to determine its type using our state-of-the-art classificatio
  </p>

  {/* Error Message */}
  {error && (
    <div className="mb-6 mx-auto max-w-lg bg-red-50 text-red-600 p-3 rounded-lg border border-red-
      {error}
    </div>
  )}

  {showCamera ? (
    <div className="mb-8 max-w-xl mx-auto border-4 border-slate-100 rounded-xl overflow-hidden sha
       <CameraCapture onCapture={handleCameraCapture} />
       <button
          onClick={() => setShowCamera(false)}
          className="w-full py-2 bg-slate-800 text-white text-sm hover:bg-slate-700 transition-col
       >
          Cancel Camera
       </button>
    </div>
  ) : (
    <div className="flex flex-col items-center space-y-6">
       {/* Preview */}
       {previewUrl && (
          <div className="mb-2">
            <img src={previewUrl} alt="Preview" className="h-40 object-contain rounded-lg border b
          </div>
       )}

       {/* Controls Row - attempting to match screenshot 'inline' look */}
       <div className="flex flex-col sm:flex-row items-center gap-4 justify-center w-full max-w-md
          <div className="relative overflow-hidden inline-block">
            <input
              type="file"
              ref={fileInputRef}
              accept="image/*"
              onChange={handleFileChange}
              className="file:mr-4 file:py-2 file:px-4
                file:rounded border border-slate-300 rounded
                file:border-0
                file:text-sm file:font-semibold
                file:bg-slate-100 file:text-slate-700
                file:cursor-pointer hover:file:bg-slate-200
                text-sm text-slate-600 cursor-pointer
              "
            />
          </div>

          <button
              onClick={handleAnalyze}
              disabled={!selectedFile || isAnalyzing}
              className={`
                px-6 py-2 rounded font-semibold text-white shadow-sm transition-colors min-w-[100p
                ${!selectedFile || isAnalyzing
                  ? 'bg-[#e74c3c]/50 cursor-not-allowed'
                  : 'bg-[#e74c3c] hover:bg-[#c0392b]'}
              `}
```

```
                    >
                        {isAnalyzing ? (
                          <Loader2 className="w-5 h-5 animate-spin mx-auto" />
                        ) : (
                          'Predict'
                        )}
                    </button>
                </div>

                {/* Camera Option */}
                <div className="pt-2">
                    <button
                        onClick={() => {
                            setShowCamera(true);
                            setPreviewUrl(null);
                            setSelectedFile(null);
                        }}
                        className="flex items-center gap-2 text-slate-500 hover:text-[#e74c3c] transition-co
                    >
                        <Camera className="w-4 h-4" />
                        <span>Or capture using camera</span>
                    </button>
                </div>
              </div>
            )}
          </section>
        </div>
      </div>
    </div>
  );
};

export default AnalysisView;
```

File: components/CameraCapture.tsx

```
import React, { useRef, useState, useEffect } from 'react';
import { Camera, X, Zap, ZapOff, ZoomIn, ZoomOut, ScanEye } from 'lucide-react';

interface CameraCaptureProps {
  onCapture: (file: File) => void;
}

const CameraCapture: React.FC<CameraCaptureProps> = ({ onCapture }) => {
  const videoRef = useRef<HTMLVideoElement>(null);
  const canvasRef = useRef<HTMLCanvasElement>(null);
  const [isStreaming, setIsStreaming] = useState(false);
  const [error, setError] = useState<string | null>(null);

  // Camera state
  const [stream, setStream] = useState<MediaStream | null>(null);
  const [capabilities, setCapabilities] = useState<MediaTrackCapabilities | null>(null);
  const [isTorchOn, setIsTorchOn] = useState(false);
  const [zoom, setZoom] = useState<number>(1);
  const [focusMode, setFocusMode] = useState<'auto' | 'manual'>('auto');
  const [focusDistance, setFocusDistance] = useState<number>(0);

  const startCamera = async () => {
    try {
      const mediaStream = await navigator.mediaDevices.getUserMedia({
        video: {
          facingMode: { ideal: 'environment' },
          width: { ideal: 1920 },
          height: { ideal: 1080 }
        }
      });

      if (videoRef.current) {
        videoRef.current.srcObject = mediaStream;
        setStream(mediaStream);
        setIsStreaming(true);
        setError(null);

        // Analyze capabilities
        const track = mediaStream.getVideoTracks()[0];
        if (track.getCapabilities) {
          const caps = track.getCapabilities();
          setCapabilities(caps);
          const settings = track.getSettings();

          // Init Zoom
          if ('zoom' in settings) {
             setZoom((settings as any).zoom as number);
          } else if ((caps as any).zoom) {
             setZoom((caps as any).zoom.min);
          }

          // Init Focus
          if ('focusMode' in settings) {
             setFocusMode(((settings as any).focusMode === 'manual') ? 'manual' : 'auto');
          }
          if ('focusDistance' in settings) {
             setFocusDistance((settings as any).focusDistance as number);
          }
        }
      }
    } catch (err) {
      console.error("Error accessing camera:", err);
      setError("Unable to access camera. Please check permissions.");
    }
```

```
};

const stopCamera = () => {
  if (stream) {
    stream.getTracks().forEach(track => {
      // Try to turn off torch before stopping
      if (isTorchOn) {
          try {
              track.applyConstraints({ advanced: [{ torch: false }] } as any).catch(() => {});
          } catch(e) {}
      }
      track.stop();
    });
    setStream(null);
  }
  if (videoRef.current) {
    videoRef.current.srcObject = null;
  }
  setIsStreaming(false);
  setIsTorchOn(false);
};

useEffect(() => {
  return () => {
    stopCamera();
  };
}, []);

const captureImage = () => {
  if (videoRef.current && canvasRef.current) {
    const context = canvasRef.current.getContext('2d');
    if (context) {
      canvasRef.current.width = videoRef.current.videoWidth;
      canvasRef.current.height = videoRef.current.videoHeight;
      context.drawImage(videoRef.current, 0, 0);
      canvasRef.current.toBlob((blob) => {
        if (blob) {
          const file = new File([blob], "camera-capture.png", { type: "image/png" });
          onCapture(file);
          stopCamera();
        }
      }, 'image/png');
    }
  }
};

// Hardware Controls
const toggleTorch = async () => {
  if (!stream) return;
  const track = stream.getVideoTracks()[0];
  const newStatus = !isTorchOn;
  try {
    await track.applyConstraints({ advanced: [{ torch: newStatus }] } as any);
    setIsTorchOn(newStatus);
  } catch (e) {
    console.error("Flash/Torch not supported", e);
  }
};

const handleZoom = async (e: React.ChangeEvent<HTMLInputElement>) => {
  const newZoom = parseFloat(e.target.value);
  setZoom(newZoom);
  if (!stream) return;
  const track = stream.getVideoTracks()[0];
  try {
    await track.applyConstraints({ advanced: [{ zoom: newZoom }] } as any);
  } catch (e) {
    console.error("Zoom failed", e);
```

```
  }
};

const toggleFocusMode = async () => {
  if (!stream || !capabilities) return;
  const track = stream.getVideoTracks()[0];
  // Usually 'continuous' is the auto-focus mode for video
  const newMode = focusMode === 'auto' ? 'manual' : 'continuous';
  try {
     await track.applyConstraints({ advanced: [{ focusMode: newMode }] } as any);
     setFocusMode(newMode === 'manual' ? 'manual' : 'auto');
  } catch (e) {
     console.error("Focus mode switch failed", e);
  }
};

const handleFocusDistance = async (e: React.ChangeEvent<HTMLInputElement>) => {
  const dist = parseFloat(e.target.value);
  setFocusDistance(dist);
  if (!stream) return;
  const track = stream.getVideoTracks()[0];
  try {
      await track.applyConstraints({ advanced: [{ focusMode: 'manual', focusDistance: dist }] } as any);
      setFocusMode('manual');
  } catch(e) { console.error("Focus distance failed", e); }
}

if (error) {
  return (
    <div className="h-64 bg-slate-100 rounded-xl flex flex-col items-center justify-center text-slate-500
      <p>{error}</p>
      <button onClick={startCamera} className="mt-4 px-4 py-2 bg-blue-600 text-white rounded-lg hover:bg-b
          Retry
      </button>
    </div>
  );
}

if (!isStreaming) {
  return (
    <div className="h-64 bg-slate-900 rounded-xl flex flex-col items-center justify-center text-white bord
      <Camera className="w-12 h-12 mb-4 text-slate-400" />
      <p className="mb-4 text-slate-300">Start camera to capture sample</p>
      <button
        onClick={startCamera}
        className="px-6 py-2 bg-blue-600 hover:bg-blue-500 rounded-full font-medium transition-colors"
      >
        Enable Camera
      </button>
    </div>
  );
}

const caps = capabilities as any;

return (
  <div className="relative h-64 md:h-80 bg-black rounded-xl overflow-hidden flex items-center justify-cent
    <video
      ref={videoRef}
      autoPlay
      playsInline
      className="absolute inset-0 w-full h-full object-cover"
    />
    <canvas ref={canvasRef} className="hidden" />

    {/* Top Right Controls */}
    <div className="absolute top-4 right-4 flex flex-col gap-2 z-20">
      {/* Flash Toggle */}
```

```
  {caps?.torch && (
    <button
      onClick={toggleTorch}
      className="p-3 bg-black/50 hover:bg-black/70 text-white rounded-full backdrop-blur-sm transition
      title="Toggle Flash"
    >
      {isTorchOn ? <Zap className="w-5 h-5 text-yellow-400 fill-current" /> : <ZapOff className="w-5 h
    </button>
  )}

  {/* Focus Mode Toggle */}
  {caps?.focusMode && caps.focusMode.includes('manual') && (
     <button
       onClick={toggleFocusMode}
       className={`p-3 rounded-full backdrop-blur-sm transition-colors ${focusMode === 'manual' ? 'bg-
       title={focusMode === 'auto' ? "Switch to Manual Focus" : "Switch to Auto Focus"}
     >
       <ScanEye className="w-5 h-5" />
     </button>
  )}
</div>

{/* Sliders Area */}
<div className="absolute bottom-24 left-6 right-6 z-20 flex flex-col gap-2">
   {/* Manual Focus Slider */}
   {focusMode === 'manual' && caps?.focusDistance && (
      <div className="flex items-center gap-3 bg-black/40 px-3 py-1.5 rounded-full backdrop-blur-sm an
          <span className="text-[10px] font-bold text-white/90 uppercase tracking-wider w-8 text-cente
          <input
              type="range"
              min={caps.focusDistance.min}
              max={caps.focusDistance.max}
              step={caps.focusDistance.step}
              value={focusDistance}
              onChange={handleFocusDistance}
              className="flex-1 h-1 bg-white/30 rounded-lg appearance-none cursor-pointer accent-yello
          />
      </div>
   )}

   {/* Zoom Slider */}
   {caps?.zoom && (
      <div className="flex items-center gap-3 bg-black/40 px-3 py-1.5 rounded-full backdrop-blur-sm">
          <ZoomOut className="w-4 h-4 text-white/90" />
          <input
              type="range"
              min={caps.zoom.min}
              max={caps.zoom.max}
              step={caps.zoom.step}
              value={zoom}
              onChange={handleZoom}
              className="flex-1 h-1 bg-white/30 rounded-lg appearance-none cursor-pointer accent-blue-
          />
          <ZoomIn className="w-4 h-4 text-white/90" />
      </div>
   )}
</div>

{/* Bottom Controls */}
<div className="absolute bottom-4 left-0 right-0 flex justify-center gap-6 z-10">
  <button
    onClick={stopCamera}
    className="p-3 bg-red-500/80 hover:bg-red-600 text-white rounded-full backdrop-blur-sm transition-
    title="Stop Camera"
  >
    <X className="w-6 h-6" />
  </button>
  <button
```

```
          onClick={captureImage}
          className="p-4 bg-white/90 hover:bg-white text-blue-600 rounded-full shadow-lg border-4 border-blu
          title="Capture Image"
        >
           <Camera className="w-8 h-8" />
        </button>
      </div>
    </div>
  );
};

export default CameraCapture;
```

File: components/Dashboard.tsx

```
import React from 'react';
import { ViewState } from '../types';
import { Microscope, Activity, Users, BookOpen, ArrowRight } from 'lucide-react';
import { BarChart, Bar, XAxis, YAxis, Tooltip, ResponsiveContainer, Cell } from 'recharts';

interface DashboardProps {
  onChangeView: (view: ViewState) => void;
}

const data = [
  { name: 'Neutrophil', count: 45 },
  { name: 'Lymphocyte', count: 30 },
  { name: 'Monocyte', count: 15 },
  { name: 'Eosinophil', count: 10 },
];

const COLORS = ['#3b82f6', '#10b981', '#f59e0b', '#ef4444'];

const Dashboard: React.FC<DashboardProps> = ({ onChangeView }) => {
  return (
    <div className="space-y-8">
      {/* Hero Section */}
      <div className="relative bg-gradient-to-r from-blue-900 to-slate-900 rounded-3xl overflow-hidden p-8 m
        <div className="relative z-10 max-w-2xl">
          <h1 className="text-3xl md:text-5xl font-bold mb-4 leading-tight">
            Advanced Blood Cell Classification
          </h1>
          <p className="text-blue-100 text-lg mb-8 leading-relaxed">
            HematoVision leverages state-of-the-art Transfer Learning and Deep Learning to provide real-time
          </p>
          <div className="flex flex-wrap gap-4">
            <button
              onClick={() => onChangeView(ViewState.ANALYSIS)}
              className="px-6 py-3 bg-blue-500 hover:bg-blue-600 text-white rounded-lg font-semibold transit
            >
              Start Diagnosis <ArrowRight className="w-5 h-5 ml-2" />
            </button>
            <button
              onClick={() => onChangeView(ViewState.EDUCATION)}
              className="px-6 py-3 bg-white/10 hover:bg-white/20 backdrop-blur-sm text-white rounded-lg font
            >
              Educational Mode
            </button>
          </div>
        </div>
        <div className="absolute right-0 bottom-0 opacity-10 pointer-events-none">
          <svg width="400" height="400" viewBox="0 0 24 24" fill="none" stroke="currentColor" strokeWidth="0
             <path d="M12 2C6.48 2 2 6.48 2 12s4.48 10 10 10 10-4.48 10-10S17.52 2 12 2zm0 18c-4.41 0-8-3.59
             <circle cx="12" cy="12" r="5" />
          </svg>
        </div>
      </div>

      {/* Stats Grid */}
      <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
        <div className="bg-white p-6 rounded-2xl shadow-sm border border-slate-200 hover:shadow-md transitio
          <div className="w-12 h-12 bg-blue-100 text-blue-600 rounded-xl flex items-center justify-center mb
            <Microscope className="w-6 h-6" />
          </div>
          <h3 className="text-lg font-semibold text-slate-800 mb-1">Scenario 1</h3>
          <p className="text-sm font-medium text-blue-600 mb-2">Automated Diagnostics</p>
          <p className="text-slate-500 text-sm">Real-time classification reducing manual workload and increa
        </div>
        <div className="bg-white p-6 rounded-2xl shadow-sm border border-slate-200 hover:shadow-md transitio
```

```
          <div className="w-12 h-12 bg-green-100 text-green-600 rounded-xl flex items-center justify-center
            <Activity className="w-6 h-6" />
          </div>
          <h3 className="text-lg font-semibold text-slate-800 mb-1">Scenario 2</h3>
          <p className="text-sm font-medium text-green-600 mb-2">Remote Consultation</p>
          <p className="text-slate-500 text-sm">Facilitating telemedicine by allowing remote upload and inst
        </div>

        <div className="bg-white p-6 rounded-2xl shadow-sm border border-slate-200 hover:shadow-md transitio
          <div className="w-12 h-12 bg-purple-100 text-purple-600 rounded-xl flex items-center justify-cente
            <BookOpen className="w-6 h-6" />
          </div>
          <h3 className="text-lg font-semibold text-slate-800 mb-1">Scenario 3</h3>
          <p className="text-sm font-medium text-purple-600 mb-2">Medical Training</p>
          <p className="text-slate-500 text-sm">Interactive educational tools with instant feedback on blood
        </div>
      </div>

      {/* Chart Section */}
      <div className="bg-white p-8 rounded-2xl shadow-sm border border-slate-200">
        <div className="flex justify-between items-center mb-6">
          <h3 className="text-lg font-bold text-slate-800">Common Cell Distribution (Dataset)</h3>
          <span className="text-sm text-slate-500 bg-slate-100 px-3 py-1 rounded-full">N = 12,000 Images</sp
        </div>
        <div className="h-64 w-full">
          <ResponsiveContainer width="100%" height="100%">
            <BarChart data={data} layout="vertical" margin={{ top: 5, right: 30, left: 40, bottom: 5 }}>
              <XAxis type="number" />
              <YAxis dataKey="name" type="category" width={100} tick={{fontSize: 12}} />
              <Tooltip cursor={{fill: 'transparent'}} contentStyle={{ borderRadius: '8px', border: 'none', b
              <Bar dataKey="count" radius={[0, 4, 4, 0]} barSize={32}>
                {data.map((entry, index) => (
                  <Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
                ))}
              </Bar>
            </BarChart>
          </ResponsiveContainer>
        </div>
      </div>
    </div>
  );
};

export default Dashboard;
```

File: components/EducationView.tsx

```
import React, { useState } from 'react';
import { generateQuiz } from '../services/hospitalService';
import { Loader2, Check, X, ArrowRight, BookOpen } from 'lucide-react';

const EducationView: React.FC = () => {
  const [loading, setLoading] = useState(false);
  const [quizData, setQuizData] = useState<any | null>(null);
  const [selectedOption, setSelectedOption] = useState<string | null>(null);
  const [showExplanation, setShowExplanation] = useState(false);

  const loadNewQuestion = async () => {
    setLoading(true);
    setQuizData(null);
    setSelectedOption(null);
    setShowExplanation(false);
    try {
      const data = await generateQuiz();
      setQuizData(data);
    } catch (e) {
      console.error(e);
    } finally {
      setLoading(false);
    }
  };

  const handleOptionSelect = (option: string) => {
    if (showExplanation) return;
    setSelectedOption(option);
    setShowExplanation(true);
  };

  React.useEffect(() => {
    loadNewQuestion();
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, []);

  return (
    <div className="max-w-3xl mx-auto">
      <div className="mb-8">
        <h2 className="text-2xl font-bold text-slate-800 flex items-center">
          <BookOpen className="w-8 h-8 mr-3 text-purple-600" />
          Morphology Knowledge Check
        </h2>
        <p className="text-slate-500 mt-2">
          Test your knowledge on blood cell types, features, and clinical significance. Questions are genera
        </p>
      </div>

      <div className="bg-white rounded-2xl shadow-sm border border-slate-200 overflow-hidden min-h-[400px]">
        {loading ? (
          <div className="flex flex-col items-center justify-center h-[400px]">
            <Loader2 className="w-12 h-12 text-purple-600 animate-spin mb-4" />
            <p className="text-slate-600 font-medium">Generating new scenario...</p>
          </div>
        ) : quizData ? (
          <div className="p-8">
            <div className="mb-8">
              <span className="inline-block px-3 py-1 bg-purple-100 text-purple-700 rounded-full text-xs fon
                Question
              </span>
              <h3 className="text-xl md:text-2xl font-semibold text-slate-800 leading-snug">
                {quizData.question}
              </h3>
            </div>
```

```
            <div className="space-y-4 mb-8">
              {quizData.options.map((option: string, idx: number) => {
                const isSelected = selectedOption === option;
                const isCorrect = option === quizData.correctAnswer;

                let containerClass = "border-slate-200 hover:border-purple-300 hover:bg-purple-50";
                if (showExplanation) {
                   if (isCorrect) containerClass = "border-green-500 bg-green-50 text-green-900";
                   else if (isSelected && !isCorrect) containerClass = "border-red-500 bg-red-50 text-red-90
                   else containerClass = "border-slate-200 opacity-50";
                } else if (isSelected) {
                   containerClass = "border-purple-500 bg-purple-50";
                }

                return (
                  <button
                    key={idx}
                    onClick={() => handleOptionSelect(option)}
                    disabled={showExplanation}
                    className={`w-full text-left p-4 rounded-xl border-2 transition-all flex justify-between
                  >
                    <span className="font-medium text-lg">{option}</span>
                    {showExplanation && isCorrect && <Check className="w-6 h-6 text-green-600" />}
                    {showExplanation && isSelected && !isCorrect && <X className="w-6 h-6 text-red-600" />}
                  </button>
                );
              })}
            </div>

            {showExplanation && (
              <div className="animate-fadeIn">
                <div className="bg-blue-50 p-6 rounded-xl border border-blue-100 mb-8">
                  <h4 className="font-semibold text-blue-900 mb-2">Explanation</h4>
                  <p className="text-blue-800 leading-relaxed">{quizData.explanation}</p>
                </div>
                <div className="flex justify-end">
                   <button
                    onClick={loadNewQuestion}
                    className="px-6 py-3 bg-slate-900 text-white rounded-lg hover:bg-slate-800 transition-co
                  >
                    Next Question <ArrowRight className="w-4 h-4 ml-2" />
                  </button>
                </div>
              </div>
            )}
          </div>
        ) : (
          <div className="p-8 text-center text-slate-500">
            Failed to load quiz. Please try again.
            <button onClick={loadNewQuestion} className="block mx-auto mt-4 text-purple-600 hover:underline"
          </div>
        )}
      </div>
    </div>
  );
};

export default EducationView;
```

File: components/HelpView.tsx

```
import React, { useState, useEffect, useRef } from 'react';
import { Play, Pause, SkipBack, SkipForward, LayoutDashboard, Microscope, GraduationCap, CheckCircle, HelpCi

const HelpView: React.FC = () => {
  const [isPlaying, setIsPlaying] = useState(false);
  const [currentStep, setCurrentStep] = useState(0);
  const progressRef = useRef<number>(0);

  const steps = [
    {
      title: "Welcome to HematoVision",
      description: "Your advanced AI assistant for blood cell morphology classification and analysis.",
      icon: Activity,
      color: "bg-red-500"
    },
    {
      title: "Dashboard Overview",
      description: "Monitor usage statistics, quick access to all modes, and view system status at a glance.
      icon: LayoutDashboard,
      color: "bg-blue-500"
    },
    {
      title: "Diagnostic Analysis",
      description: "Upload blood smear images or use the camera. The AI instantly identifies cell types with
      icon: Microscope,
      color: "bg-purple-500"
    },
    {
      title: "Results & Morphology",
      description: "Get detailed reports including cell type, confidence score, morphology description, and
      icon: CheckCircle,
      color: "bg-green-500"
    },
    {
      title: "Education Mode",
      description: "Test your knowledge with AI-generated quizzes to improve your diagnostic skills.",
      icon: GraduationCap,
      color: "bg-orange-500"
    }
  ];

  // Auto-play logic
  useEffect(() => {
    let interval: any;
    if (isPlaying) {
      interval = setInterval(() => {
        setCurrentStep((prev) => {
          if (prev >= steps.length - 1) {
            setIsPlaying(false);
            return 0; // Reset or stay at end? Let's loop or stop. Stopping is better for a manual.
          }
          return prev + 1;
        });
      }, 3000); // 3 seconds per slide
    }
    return () => clearInterval(interval);
  }, [isPlaying, steps.length]);

  const togglePlay = () => setIsPlaying(!isPlaying);

  const handleStepClick = (index: number) => {
    setCurrentStep(index);
    setIsPlaying(false); // Pause if user manually interacts
  };
```

```
const CurrentIcon = steps[currentStep].icon;

return (
  <div className="max-w-4xl mx-auto space-y-8">
    <div className="mb-6">
      <h2 className="text-3xl font-bold text-slate-800 flex items-center">
        <HelpCircle className="w-8 h-8 mr-3 text-blue-600" />
        User Manual & Guide
      </h2>
      <p className="text-slate-500 mt-2">
        Learn how to operate the HematoVision application effectively.
      </p>
    </div>

    {/* Video Player Simulation */}
    <div className="bg-slate-900 rounded-2xl overflow-hidden shadow-2xl aspect-video flex flex-col relativ

      {/* Screen Content */}
      <div className="flex-1 flex flex-col items-center justify-center p-8 text-center relative overflow-h

         {/* Background Animation Element */}
         <div className={`absolute inset-0 opacity-20 transition-colors duration-500 ${steps[currentStep].
         <div className="absolute inset-0 bg-gradient-to-t from-slate-900 via-transparent to-transparent"

         {/* Main Icon Animation */}
         <div className="relative z-10 mb-8 transform transition-all duration-500 ease-out scale-100">
            <div className={`w-24 h-24 rounded-3xl flex items-center justify-center shadow-lg mb-6 mx-auto
               <CurrentIcon className="w-12 h-12 text-white" />
            </div>
            <h3 className="text-3xl font-bold text-white mb-4 tracking-tight">{steps[currentStep].title}</
            <p className="text-slate-300 text-lg max-w-xl mx-auto leading-relaxed">
              {steps[currentStep].description}
            </p>
         </div>
      </div>

      {/* Player Controls */}
      <div className="bg-slate-800/80 backdrop-blur-md p-4 border-t border-slate-700">
         {/* Progress Bar */}
         <div className="w-full h-1.5 bg-slate-700 rounded-full mb-4 overflow-hidden">
            <div
              className="h-full bg-blue-500 transition-all duration-300 ease-linear"
              style={{ width: `${((currentStep + 1) / steps.length) * 100}%` }}
            />
         </div>

         <div className="flex items-center justify-between">
            <div className="flex items-center space-x-4">
               <button
                 onClick={togglePlay}
                 className="w-10 h-10 bg-white text-slate-900 rounded-full flex items-center justify-cente
               >
                  {isPlaying ? <Pause className="w-5 h-5 fill-current" /> : <Play className="w-5 h-5 fill-
               </button>
               <div className="text-slate-400 text-sm font-medium">
                  {Math.floor((currentStep + 1) * 3)}s / {steps.length * 3}s
               </div>
            </div>

            <div className="flex items-center space-x-2">
               <button
                  onClick={() => setCurrentStep(Math.max(0, currentStep - 1))}
                  className="p-2 text-slate-400 hover:text-white transition-colors"
               >
                  <SkipBack className="w-5 h-5" />
               </button>
               <button
                  onClick={() => setCurrentStep(Math.min(steps.length - 1, currentStep + 1))}
```

```
                  className="p-2 text-slate-400 hover:text-white transition-colors"
                >
                  <SkipForward className="w-5 h-5" />
                </button>
              </div>
            </div>
          </div>
        </div>

        {/* Written Instructions */}
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6 mt-8">
           <div className="bg-white p-6 rounded-xl border border-slate-200 shadow-sm">
              <h3 className="font-bold text-slate-800 mb-4 flex items-center">
                 <Microscope className="w-5 h-5 mr-2 text-purple-600" />
                 How to Analyze an Image
              </h3>
              <ol className="space-y-4 text-slate-600 text-sm">
                 <li className="flex gap-3">
                    <span className="flex-shrink-0 w-6 h-6 bg-slate-100 text-slate-600 rounded-full flex items
                    <span>Navigate to the <strong>Diagnostic Analysis</strong> tab from the sidebar.</span>
                 </li>
                 <li className="flex gap-3">
                    <span className="flex-shrink-0 w-6 h-6 bg-slate-100 text-slate-600 rounded-full flex items
                    <span>Click "Upload Blood Sample Image" to select a file from your device, or drag and dro
                 </li>
                 <li className="flex gap-3">
                    <span className="flex-shrink-0 w-6 h-6 bg-slate-100 text-slate-600 rounded-full flex items
                    <span>Alternatively, use the "Capture using camera" option for real-time analysis.</span>
                 </li>
                 <li className="flex gap-3">
                    <span className="flex-shrink-0 w-6 h-6 bg-slate-100 text-slate-600 rounded-full flex items
                    <span>Click the red <strong>Predict</strong> button and wait for the AI to process the ima
                 </li>
              </ol>
           </div>

           <div className="bg-white p-6 rounded-xl border border-slate-200 shadow-sm">
              <h3 className="font-bold text-slate-800 mb-4 flex items-center">
                 <GraduationCap className="w-5 h-5 mr-2 text-orange-600" />
                 Using Education Mode
              </h3>
              <ol className="space-y-4 text-slate-600 text-sm">
                 <li className="flex gap-3">
                    <span className="flex-shrink-0 w-6 h-6 bg-slate-100 text-slate-600 rounded-full flex items
                    <span>Select <strong>Education & Training</strong> from the main menu.</span>
                 </li>
                 <li className="flex gap-3">
                    <span className="flex-shrink-0 w-6 h-6 bg-slate-100 text-slate-600 rounded-full flex items
                    <span>Read the AI-generated scenario or question carefully.</span>
                 </li>
                 <li className="flex gap-3">
                    <span className="flex-shrink-0 w-6 h-6 bg-slate-100 text-slate-600 rounded-full flex items
                    <span>Select an answer from the multiple-choice options.</span>
                 </li>
                 <li className="flex gap-3">
                    <span className="flex-shrink-0 w-6 h-6 bg-slate-100 text-slate-600 rounded-full flex items
                    <span>Review the detailed explanation provided for the correct answer.</span>
                 </li>
              </ol>
           </div>
        </div>
      </div>
    );
};

export default HelpView;
```

File: components/ImageUpload.tsx

```
import React, { useCallback } from 'react';
import { Upload, X } from 'lucide-react';

interface ImageUploadProps {
  onImageSelected: (file: File) => void;
  selectedImage: string | null;
  onClear: () => void;
}

const ImageUpload: React.FC<ImageUploadProps> = ({ onImageSelected, selectedImage, onClear }) => {
  const handleFileChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    if (event.target.files && event.target.files[0]) {
      onImageSelected(event.target.files[0]);
    }
  };

  const handleDrop = useCallback(
    (e: React.DragEvent<HTMLDivElement>) => {
      e.preventDefault();
      e.stopPropagation();
      if (e.dataTransfer.files && e.dataTransfer.files[0]) {
        onImageSelected(e.dataTransfer.files[0]);
      }
    },
    [onImageSelected]
  );

  const handleDragOver = (e: React.DragEvent<HTMLDivElement>) => {
    e.preventDefault();
    e.stopPropagation();
  };

  if (selectedImage) {
    return (
      <div className="relative rounded-xl overflow-hidden shadow-lg border border-slate-200 bg-slate-900 gro
        <img
          src={selectedImage}
          alt="Selected"
          className="max-w-full max-h-full object-contain"
        />
        <button
          onClick={onClear}
          className="absolute top-4 right-4 p-2 bg-red-500 hover:bg-red-600 text-white rounded-full shadow-m
        >
          <X className="w-5 h-5" />
        </button>
      </div>
    );
  }

  return (
    <div
      onDrop={handleDrop}
      onDragOver={handleDragOver}
      className="border-2 border-dashed border-slate-300 rounded-xl p-8 md:p-12 text-center hover:border-blu
    >
      <input
        type="file"
        accept="image/*"
        onChange={handleFileChange}
        className="hidden"
        id="file-upload"
      />
      <label htmlFor="file-upload" className="cursor-pointer flex flex-col items-center w-full h-full justif
```

```
        <div className="w-16 h-16 bg-blue-100 text-blue-600 rounded-full flex items-center justify-center mb
          <Upload className="w-8 h-8" />
        </div>
        <h3 className="text-lg font-semibold text-slate-800 mb-2">Upload Blood Sample Image</h3>
        <p className="text-slate-500 text-sm max-w-xs mx-auto">
          Drag and drop your microscopy image here, or click to browse files.
        </p>
        <p className="text-xs text-slate-400 mt-4">Supported formats: JPG, PNG</p>
      </label>
    </div>
  );
};

export default ImageUpload;
```

File: components/Layout.tsx

```
import React from 'react';
import { ViewState } from '../types';
import { LayoutDashboard, Microscope, GraduationCap, Settings, Menu, X, Activity, LineChart, Presentation, H

interface LayoutProps {
  currentView: ViewState;
  onViewChange: (view: ViewState) => void;
  children: React.ReactNode;
}

const Layout: React.FC<LayoutProps> = ({ currentView, onViewChange, children }) => {
  const [isMobileMenuOpen, setIsMobileMenuOpen] = React.useState(false);

  const NavItem = ({ view, icon: Icon, label }: { view: ViewState; icon: any; label: string }) => (
    <button
      onClick={() => {
        onViewChange(view);
        setIsMobileMenuOpen(false);
      }}
      className={`flex items-center w-full px-4 py-3 mb-2 rounded-lg transition-colors ${
        currentView === view
          ? 'bg-blue-600 text-white shadow-md'
          : 'text-slate-600 hover:bg-slate-100'
      }`}
    >
      <Icon className="w-5 h-5 mr-3" />
      <span className="font-medium">{label}</span>
    </button>
  );

  return (
    <div className="flex h-screen bg-slate-50 overflow-hidden">
      {/* Sidebar - Desktop */}
      <aside className="hidden md:flex flex-col w-64 bg-white border-r border-slate-200">
        <div className="flex items-center px-6 py-6 border-b border-slate-100">
          <div className="w-8 h-8 bg-red-500 rounded-lg flex items-center justify-center mr-3 shadow-sm">
            <Activity className="w-5 h-5 text-white" />
          </div>
          <h1 className="text-xl font-bold text-slate-800">HematoVision</h1>
        </div>
        <nav className="flex-1 px-4 py-6 overflow-y-auto">
          <NavItem view={ViewState.DASHBOARD} icon={LayoutDashboard} label="Dashboard" />
          <NavItem view={ViewState.ANALYSIS} icon={Microscope} label="Diagnostic Analysis" />
          <NavItem view={ViewState.MODEL_EVALUATION} icon={LineChart} label="Model Evaluation" />
          <NavItem view={ViewState.EDUCATION} icon={GraduationCap} label="Education & Training" />
          <div className="my-4 border-t border-slate-100"></div>
          <NavItem view={ViewState.PRESENTATION} icon={Presentation} label="Project Presentation" />
          <NavItem view={ViewState.HELP} icon={HelpCircle} label="User Manual" />
        </nav>
        <div className="p-4 border-t border-slate-100">
          <div className="flex items-center px-4 py-2 text-sm text-slate-500">
             <span className="w-2 h-2 bg-green-500 rounded-full mr-2"></span>
             System Online
          </div>
        </div>
      </aside>

      {/* Main Content */}
      <div className="flex-1 flex flex-col min-w-0 overflow-hidden">
        {/* Mobile Header */}
        <header className="md:hidden flex items-center justify-between px-4 py-3 bg-white border-b border-sl
           <div className="flex items-center">
            <div className="w-8 h-8 bg-red-500 rounded-lg flex items-center justify-center mr-3">
              <Activity className="w-5 h-5 text-white" />
```

```
            </div>
            <h1 className="text-lg font-bold text-slate-800">HematoVision</h1>
          </div>
          <button onClick={() => setIsMobileMenuOpen(!isMobileMenuOpen)} className="p-2 text-slate-600">
            {isMobileMenuOpen ? <X /> : <Menu />}
          </button>
        </header>

        {/* Mobile Menu */}
        {isMobileMenuOpen && (
          <div className="md:hidden absolute top-16 left-0 right-0 z-50 bg-white border-b border-slate-200 s
             <nav>
              <NavItem view={ViewState.DASHBOARD} icon={LayoutDashboard} label="Dashboard" />
              <NavItem view={ViewState.ANALYSIS} icon={Microscope} label="Diagnostic Analysis" />
              <NavItem view={ViewState.MODEL_EVALUATION} icon={LineChart} label="Model Evaluation" />
              <NavItem view={ViewState.EDUCATION} icon={GraduationCap} label="Education & Training" />
              <NavItem view={ViewState.PRESENTATION} icon={Presentation} label="Project Presentation" />
              <NavItem view={ViewState.HELP} icon={HelpCircle} label="User Manual" />
            </nav>
          </div>
        )}

        <main className="flex-1 overflow-auto p-4 md:p-8">
          <div className="max-w-6xl mx-auto">
            {children}
          </div>
        </main>
      </div>
    </div>
  );
};

export default Layout;
```

File: components/ModelEvaluation.tsx

```
import React from 'react';
import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, Legend, ResponsiveContainer } from 'recharts
import { Brain, FileCheck, Database, Layers, Server } from 'lucide-react';

const accuracyData = [
  { epoch: 0, train: 0.36, val: 0.53 },
  { epoch: 1, train: 0.52, val: 0.64 },
  { epoch: 2, train: 0.65, val: 0.75 },
  { epoch: 3, train: 0.76, val: 0.85 },
  { epoch: 4, train: 0.85, val: 0.89 },
  { epoch: 5, train: 0.89, val: 0.88 },
];

const lossData = [
  { epoch: 0, train: 1.6, val: 1.05 },
  { epoch: 1, train: 1.1, val: 0.83 },
  { epoch: 2, train: 0.83, val: 0.60 },
  { epoch: 3, train: 0.58, val: 0.38 },
  { epoch: 4, train: 0.39, val: 0.29 },
  { epoch: 5, train: 0.28, val: 0.30 },
];

const classificationReport = [
  { class: 'Eosinophil', precision: 0.82, recall: 0.81, f1: 0.82, support: 725 },
  { class: 'Lymphocyte', precision: 0.90, recall: 0.99, f1: 0.94, support: 762 },
  { class: 'Monocyte', precision: 0.98, recall: 0.96, f1: 0.97, support: 759 },
  { class: 'Neutrophil', precision: 0.87, recall: 0.80, f1: 0.83, support: 742 },
];

const confusionMatrix = [
  { actual: 'Eosinophil', preds: [590, 52, 5, 78] },
  { actual: 'Lymphocyte', preds: [4, 757, 0, 1] },
  { actual: 'Monocyte', preds: [1, 17, 729, 12] },
  { actual: 'Neutrophil', preds: [124, 16, 10, 592] },
];

const classes = ['EOSINOPHIL', 'LYMPHOCYTE', 'MONOCYTE', 'NEUTROPHIL'];

const ModelEvaluation: React.FC = () => {
  return (
    <div className="space-y-8 animate-fadeIn">
      {/* Enhanced Header with Prominent Model Info */}
      <div className="bg-white rounded-2xl shadow-sm border border-slate-200 overflow-hidden">
        <div className="p-8 border-b border-slate-100 bg-slate-50/50">
           <div className="flex items-center mb-8">
              <div className="p-3 bg-blue-600 rounded-xl mr-4 shadow-lg shadow-blue-600/20">
                 <Brain className="w-8 h-8 text-white" />
              </div>
              <div>
                 <h2 className="text-3xl font-bold text-slate-900">MobileNet V2 Performance</h2>
                 <p className="text-slate-500 text-lg">Deep Learning Classification Evaluation</p>
              </div>
           </div>

           <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
              <div className="bg-white p-5 rounded-xl border border-slate-200 shadow-sm flex items-start spa
                 <div className="p-2 bg-purple-100 rounded-lg text-purple-600">
                    <Layers className="w-6 h-6" />
                 </div>
                 <div>
                    <p className="text-sm font-semibold text-slate-500 uppercase tracking-wide">Architecture
                    <p className="text-xl font-bold text-slate-900">MobileNet V2</p>
                    <p className="text-sm text-slate-400">Transfer Learning</p>
                 </div>
```

```
        </div>

        <div className="bg-white p-5 rounded-xl border border-slate-200 shadow-sm flex items-start spa
           <div className="p-2 bg-indigo-100 rounded-lg text-indigo-600">
              <Database className="w-6 h-6" />
           </div>
           <div>
              <p className="text-sm font-semibold text-slate-500 uppercase tracking-wide">Total Datase
              <p className="text-xl font-bold text-slate-900">12,000 Images</p>
              <p className="text-sm text-slate-400">Annotated Blood Cells</p>
           </div>
        </div>

        <div className="bg-white p-5 rounded-xl border border-slate-200 shadow-sm flex items-start spa
           <div className="p-2 bg-orange-100 rounded-lg text-orange-600">
              <Server className="w-6 h-6" />
           </div>
           <div>
              <p className="text-sm font-semibold text-slate-500 uppercase tracking-wide">Test Split</
              <p className="text-xl font-bold text-slate-900">2,988 Images</p>
              <p className="text-sm text-slate-400">Unseen Validation Data</p>
           </div>
        </div>
     </div>
  </div>
</div>

{/* Metrics Cards */}
<div className="grid grid-cols-1 md:grid-cols-3 gap-6">
  <div className="bg-white p-6 rounded-xl border border-slate-200 shadow-sm flex flex-col items-center
      <h3 className="text-slate-500 font-medium mb-1">Overall Accuracy</h3>
      <span className="text-4xl font-bold text-blue-600">89.3%</span>
  </div>
  <div className="bg-white p-6 rounded-xl border border-slate-200 shadow-sm flex flex-col items-center
      <h3 className="text-slate-500 font-medium mb-1">Training Images</h3>
      <span className="text-4xl font-bold text-slate-800">9,012</span>
  </div>
  <div className="bg-white p-6 rounded-xl border border-slate-200 shadow-sm flex flex-col items-center
      <h3 className="text-slate-500 font-medium mb-1">Macro Avg F1-Score</h3>
      <span className="text-4xl font-bold text-green-600">0.89</span>
  </div>
</div>

{/* Charts Row */}
<div className="grid grid-cols-1 lg:grid-cols-2 gap-8">
  {/* Accuracy Chart */}
  <div className="bg-white p-6 rounded-2xl shadow-sm border border-slate-200">
    <h3 className="text-lg font-bold text-slate-800 mb-6">Model Accuracy (Train vs Val)</h3>
    <div className="h-64 w-full">
      <ResponsiveContainer width="100%" height="100%">
        <LineChart data={accuracyData}>
          <CartesianGrid strokeDasharray="3 3" vertical={false} stroke="#e2e8f0" />
          <XAxis dataKey="epoch" label={{ value: 'Epoch', position: 'insideBottom', offset: -5 }} />
          <YAxis domain={[0.3, 1.0]} label={{ value: 'Accuracy', angle: -90, position: 'insideLeft' }}
          <Tooltip
              contentStyle={{ borderRadius: '8px', border: '1px solid #e2e8f0', boxShadow: '0 4px 6px
          />
          <Legend verticalAlign="top" height={36} />
          <Line type="monotone" dataKey="train" stroke="#0284c7" strokeWidth={2} name="Train Accuracy"
          <Line type="monotone" dataKey="val" stroke="#f97316" strokeWidth={2} name="Validation Accura
        </LineChart>
      </ResponsiveContainer>
    </div>
  </div>

  {/* Loss Chart */}
  <div className="bg-white p-6 rounded-2xl shadow-sm border border-slate-200">
    <h3 className="text-lg font-bold text-slate-800 mb-6">Model Loss (Train vs Val)</h3>
```

```
    <div className="h-64 w-full">
      <ResponsiveContainer width="100%" height="100%">
        <LineChart data={lossData}>
          <CartesianGrid strokeDasharray="3 3" vertical={false} stroke="#e2e8f0" />
          <XAxis dataKey="epoch" label={{ value: 'Epoch', position: 'insideBottom', offset: -5 }} />
          <YAxis label={{ value: 'Loss', angle: -90, position: 'insideLeft' }} />
          <Tooltip
              contentStyle={{ borderRadius: '8px', border: '1px solid #e2e8f0', boxShadow: '0 4px 6px
          />
          <Legend verticalAlign="top" height={36} />
          <Line type="monotone" dataKey="train" stroke="#0284c7" strokeWidth={2} name="Train Loss" dot
          <Line type="monotone" dataKey="val" stroke="#f97316" strokeWidth={2} name="Validation Loss"
        </LineChart>
      </ResponsiveContainer>
    </div>
  </div>
</div>

{/* Confusion Matrix */}
<div className="bg-white p-6 rounded-2xl shadow-sm border border-slate-200">
   <h3 className="text-lg font-bold text-slate-800 mb-6">Confusion Matrix</h3>
   <div className="overflow-x-auto">
      <div className="min-w-[600px] flex flex-col items-center">
          {/* X Labels */}
          <div className="flex mb-2 ml-24">
             {classes.map(c => (
               <div key={c} className="w-32 text-center text-xs font-semibold text-slate-500 uppercase
             ))}
          </div>

          {confusionMatrix.map((row, idx) => (
             <div key={row.actual} className="flex items-center mb-1">
                {/* Y Label */}
                <div className="w-24 text-right pr-4 text-xs font-semibold text-slate-500 uppercase">{

                {/* Cells */}
                {row.preds.map((val, colIdx) => {
                    // Calculate heat intensity 0-1 based on max value approx 760
                    const intensity = Math.min(val / 760, 1);
                    const isHigh = intensity > 0.5;
                    return (
                      <div
                          key={colIdx}
                          className="w-32 h-16 flex items-center justify-center text-sm font-medium bo
                          style={{ backgroundColor: `rgba(59, 130, 246, ${Math.max(intensity, 0.05)})`
                      >
                          {val}
                      </div>
                    );
                })}
             </div>
          ))}

          <div className="mt-4 text-sm text-slate-400">Predicted Label (X-Axis) vs Actual Label (Y-Axi
      </div>
   </div>
</div>

{/* Classification Report Table */}
<div className="bg-white p-6 rounded-2xl shadow-sm border border-slate-200">
  <div className="flex items-center justify-between mb-6">
      <h3 className="text-lg font-bold text-slate-800">Classification Report</h3>
      <div className="flex items-center text-sm text-slate-500">
          <FileCheck className="w-4 h-4 mr-2" />
          <span>Test Set Results</span>
      </div>
  </div>
  <div className="overflow-x-auto">
```

```
            <table className="w-full text-left border-collapse">
                <thead>
                    <tr className="border-b border-slate-200">
                        <th className="py-3 px-4 text-sm font-bold text-slate-700">Class</th>
                        <th className="py-3 px-4 text-sm font-bold text-slate-700">Precision</th>
                        <th className="py-3 px-4 text-sm font-bold text-slate-700">Recall</th>
                        <th className="py-3 px-4 text-sm font-bold text-slate-700">F1-Score</th>
                        <th className="py-3 px-4 text-sm font-bold text-slate-700">Support</th>
                    </tr>
                </thead>
                <tbody>
                    {classificationReport.map((row) => (
                        <tr key={row.class} className="border-b border-slate-100 hover:bg-slate-50">
                            <td className="py-3 px-4 font-medium text-slate-800">{row.class}</td>
                            <td className="py-3 px-4 text-slate-600">{row.precision.toFixed(2)}</td>
                            <td className="py-3 px-4 text-slate-600">{row.recall.toFixed(2)}</td>
                            <td className="py-3 px-4 text-slate-600 font-semibold text-blue-600">{row.f1.toF
                            <td className="py-3 px-4 text-slate-600">{row.support}</td>
                        </tr>
                    ))}
                    <tr className="bg-slate-50 font-bold">
                        <td className="py-3 px-4 text-slate-800">Accuracy</td>
                        <td className="py-3 px-4"></td>
                        <td className="py-3 px-4"></td>
                        <td className="py-3 px-4 text-green-600">0.89</td>
                        <td className="py-3 px-4 text-slate-800">2988</td>
                    </tr>
                </tbody>
            </table>
        </div>
      </div>

    </div>
  );
};

export default ModelEvaluation;
```

File: components/PresentationView.tsx

```
import React, { useState } from 'react';
import { ChevronRight, ChevronLeft, User, School, BookOpen, Brain, Activity, CheckCircle } from 'lucide-reac

const slides = [
  {
    id: 'title',
    title: 'HematoVision',
    subtitle: 'Advanced Blood Cell Classification Using Transfer Learning',
    content: (
      <div className="flex flex-col items-center justify-center h-full space-y-8 mt-8">
        <div className="p-6 bg-blue-50 rounded-2xl border border-blue-100 w-full max-w-2xl">
           <div className="flex items-center space-x-4 mb-4">
              <User className="w-8 h-8 text-blue-600" />
              <div>
                <p className="text-sm text-slate-500 uppercase tracking-wide font-semibold">Presented By</p>
                <h3 className="text-2xl font-bold text-slate-800">Garlapati Manasa</h3>
                <p className="text-slate-600 font-medium">Roll Number: 22F61A4795</p>
              </div>
           </div>
        </div>

        <div className="p-6 bg-purple-50 rounded-2xl border border-purple-100 w-full max-w-2xl">
           <div className="flex items-center space-x-4 mb-4">
              <School className="w-8 h-8 text-purple-600" />
              <div>
                <p className="text-sm text-slate-500 uppercase tracking-wide font-semibold">Institution</p>
                <h3 className="text-xl font-bold text-slate-800">Siddhartha Institute of Engineering and Tec
                <p className="text-slate-600 font-medium">Puttur</p>
              </div>
           </div>
        </div>
      </div>
    )
  },
  {
    id: 'problem',
    title: 'Problem Statement',
    subtitle: 'Why Automated Classification Matters',
    content: (
      <div className="grid grid-cols-1 md:grid-cols-2 gap-8 mt-8">
        <div className="bg-red-50 p-6 rounded-xl border border-red-100">
          <h3 className="text-xl font-bold text-red-700 mb-4 flex items-center"><Activity className="mr-2" /
          <ul className="space-y-3 text-slate-700">
            <li className="flex items-start"><span className="mr-2">•</span> Time-consuming process for path
            <li className="flex items-start"><span className="mr-2">•</span> Prone to human error and fatigu
            <li className="flex items-start"><span className="mr-2">•</span> Subjective interpretation varia
            <li className="flex items-start"><span className="mr-2">•</span> Shortage of skilled hematologis
          </ul>
        </div>
        <div className="bg-green-50 p-6 rounded-xl border border-green-100">
          <h3 className="text-xl font-bold text-green-700 mb-4 flex items-center"><CheckCircle className="mr
          <ul className="space-y-3 text-slate-700">
            <li className="flex items-start"><span className="mr-2">•</span> <strong>Automated:</strong> Ins
            <li className="flex items-start"><span className="mr-2">•</span> <strong>Consistent:</strong> St
            <li className="flex items-start"><span className="mr-2">•</span> <strong>Accessible:</strong> We
            <li className="flex items-start"><span className="mr-2">•</span> <strong>Educational:</strong> H
          </ul>
        </div>
      </div>
    )
  },
  {
    id: 'methodology',
    title: 'Methodology & Tech Stack',
```

```
  subtitle: 'Leveraging Transfer Learning with MobileNet V2',
  content: (
    <div className="mt-8 space-y-6">
      <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
         <div className="bg-white p-5 rounded-lg shadow-sm border border-slate-200">
            <div className="w-10 h-10 bg-indigo-100 text-indigo-600 rounded-full flex items-center justify
               <Brain className="w-5 h-5" />
            </div>
            <h4 className="font-bold text-slate-800 mb-2">Model Architecture</h4>
            <p className="text-sm text-slate-600">MobileNet V2 pretrained on ImageNet, fine-tuned on blood
         </div>
         <div className="bg-white p-5 rounded-lg shadow-sm border border-slate-200">
            <div className="w-10 h-10 bg-blue-100 text-blue-600 rounded-full flex items-center justify-cen
               <Activity className="w-5 h-5" />
            </div>
            <h4 className="font-bold text-slate-800 mb-2">Dataset</h4>
            <p className="text-sm text-slate-600">12,500 augmented images split into Training (10k) and Te
         </div>
         <div className="bg-white p-5 rounded-lg shadow-sm border border-slate-200">
            <div className="w-10 h-10 bg-teal-100 text-teal-600 rounded-full flex items-center justify-cen
               <BookOpen className="w-5 h-5" />
            </div>
            <h4 className="font-bold text-slate-800 mb-2">Frontend Stack</h4>
            <p className="text-sm text-slate-600">React 18, TypeScript, Tailwind CSS, Lucide Icons.</p>
         </div>
      </div>

      <div className="bg-slate-900 text-white p-6 rounded-xl">
         <h4 className="font-bold text-lg mb-3">Workflow</h4>
         <div className="flex items-center justify-between text-sm md:text-base overflow-x-auto">
            <span>Image Upload</span>
            <span className="text-slate-500">→</span>
            <span>Preprocessing</span>
            <span className="text-slate-500">→</span>
            <span>MobileNet V2</span>
            <span className="text-slate-500">→</span>
            <span>Classification</span>
            <span className="text-slate-500">→</span>
            <span>Report</span>
         </div>
      </div>
    </div>
  )
},
{
  id: 'results',
  title: 'Results & Conclusion',
  subtitle: 'High Accuracy and Real-world Application',
  content: (
    <div className="mt-8 flex flex-col items-center">
       <div className="grid grid-cols-2 md:grid-cols-4 gap-4 w-full mb-8">
          <div className="bg-blue-50 p-4 rounded-lg text-center border border-blue-100">
             <div className="text-3xl font-bold text-blue-600">89.3%</div>
             <div className="text-xs font-semibold text-slate-500 uppercase mt-1">Accuracy</div>
          </div>
          <div className="bg-green-50 p-4 rounded-lg text-center border border-green-100">
             <div className="text-3xl font-bold text-green-600">0.89</div>
             <div className="text-xs font-semibold text-slate-500 uppercase mt-1">F1 Score</div>
          </div>
          <div className="bg-orange-50 p-4 rounded-lg text-center border border-orange-100">
             <div className="text-3xl font-bold text-orange-600">0.28</div>
             <div className="text-xs font-semibold text-slate-500 uppercase mt-1">Min Loss</div>
          </div>
          <div className="bg-purple-50 p-4 rounded-lg text-center border border-purple-100">
             <div className="text-3xl font-bold text-purple-600">4</div>
             <div className="text-xs font-semibold text-slate-500 uppercase mt-1">Classes</div>
          </div>
       </div>
```

```
        <div className="bg-white p-6 rounded-xl border-l-4 border-blue-500 shadow-sm w-full">
           <h3 className="text-lg font-bold text-slate-800 mb-2">Conclusion</h3>
           <p className="text-slate-600 leading-relaxed">
              HematoVision successfully demonstrates the power of Transfer Learning in medical diagnostics.
              By achieving high accuracy on a lightweight model, it offers a viable solution for assisting
              healthcare professionals and educating students in hematology.
           </p>
        </div>
      </div>
    )
  }
];

const PresentationView: React.FC = () => {
  const [currentSlide, setCurrentSlide] = useState(0);

  const nextSlide = () => {
    if (currentSlide < slides.length - 1) {
      setCurrentSlide(prev => prev + 1);
    }
  };

  const prevSlide = () => {
    if (currentSlide > 0) {
      setCurrentSlide(prev => prev - 1);
    }
  };

  return (
    <div className="flex flex-col h-[calc(100vh-8rem)]">
      {/* Slide Container */}
      <div className="flex-1 bg-white rounded-2xl shadow-xl border border-slate-200 overflow-hidden relative

        {/* Slide Header */}
        <div className="bg-slate-900 text-white p-8 pb-12">
           <h1 className="text-3xl md:text-4xl font-bold mb-2">{slides[currentSlide].title}</h1>
           <p className="text-blue-200 text-lg font-light">{slides[currentSlide].subtitle}</p>
        </div>

        {/* Slide Content */}
        <div className="flex-1 p-8 md:p-12 overflow-y-auto bg-slate-50/50">
           {slides[currentSlide].content}
        </div>

        {/* Footer/Progress */}
        <div className="bg-white border-t border-slate-100 p-4 flex justify-between items-center text-sm tex
           <div>HematoVision Project Presentation</div>
           <div>Slide {currentSlide + 1} of {slides.length}</div>
        </div>
      </div>

      {/* Navigation Controls */}
      <div className="flex justify-center items-center mt-6 gap-4">
        <button
          onClick={prevSlide}
          disabled={currentSlide === 0}
          className={`p-3 rounded-full shadow-md transition-all ${
            currentSlide === 0
              ? 'bg-slate-200 text-slate-400 cursor-not-allowed'
              : 'bg-white text-blue-600 hover:bg-blue-50 hover:scale-105'
          }`}
        >
          <ChevronLeft className="w-6 h-6" />
        </button>

        <div className="flex gap-2">
           {slides.map((_, idx) => (
             <button
```

```
              key={idx}
              onClick={() => setCurrentSlide(idx)}
              className={`w-3 h-3 rounded-full transition-all ${
                 currentSlide === idx ? 'bg-blue-600 w-6' : 'bg-slate-300 hover:bg-slate-400'
              }`}
            />
          ))}
        </div>

        <button
          onClick={nextSlide}
          disabled={currentSlide === slides.length - 1}
          className={`p-3 rounded-full shadow-md transition-all ${
            currentSlide === slides.length - 1
              ? 'bg-slate-200 text-slate-400 cursor-not-allowed'
              : 'bg-white text-blue-600 hover:bg-blue-50 hover:scale-105'
          }`}
        >
          <ChevronRight className="w-6 h-6" />
        </button>
      </div>
    </div>
  );
};

export default PresentationView;
```