Project Report: Cardiovascular Disease Risk Prediction System

Project Title:

CVD Risk Predictor — AI-powered Heart Disease Detection System

Tech Stack:

- **Frontend:** React + Vite + Tailwind CSS + Framer Motion
- **Backend:** Node.js + Express.js
- **Machine Learning Model:** ONNX Runtime (onnxruntime-node)
- **Deployment:**
  - **Frontend → Vercel**
  - **Backend → Render**

1. Project Overview

The CVD Risk Predictor is an intelligent web-based system that predicts the likelihood of a patient having cardiovascular disease.
It uses a trained machine learning model (converted into an ONNX format) to perform fast and scalable predictions on a web server.

Users enter clinical and lifestyle data such as:

- **Age**
- **Cholesterol level**
- **Resting blood pressure**
- **Heart rate**
- **ECG results, etc.**

The app then instantly provides a personalized risk level:

🟢 **Low Risk |** 🟡 **Moderate Risk |** 🔴 **High Risk**

and a probability score (0–100%).

2. System Architecture

Frontend (Vercel)

- Built using **React + Vite**
- UI powered by **TailwindCSS** and **Framer Motion**
- Provides a **step-based input form** with clean transitions
- Displays **animated results** using a gauge visualization
- Downloads **PDF report** for patient summary

- Uses **Axios** to communicate with the backend API

Backend (Render)

- Built using **Node.js + Express**
- Hosts and executes the **ONNX model** (model.onnx)
- Includes a **scaler normalization** (manual feature scaling)
- Uses onnxruntime-node for inference on the CPU
- Protects with:
  - **CORS** → allows requests from both localhost & Vercel
  - **Rate limiting** → max 200 requests/min
  - **morgan** → API logging

Machine Learning Model

- Model trained offline and exported as model.onnx
- StandardScaler normalization used with:
  - Mean & scale vectors (SCALER_MEAN, SCALER_SCALE)
- Input features:
- age, sex, cp, trestbps, chol, fbs,
- restecg, thalach, exang, oldpeak,
- slope, ca, thal

3. Deployment Setup

Frontend (Vercel)

- Framework: React (Vite)
- Deployed via GitHub repository
- Domain:
  🔗 https://cvd-frontend.vercel.app

Backend (Render)

- Environment: Node.js
- Deployed from GitHub repository
- Domain:
  🔗 https://cvd-backend-w8p6.onrender.com

4. Communication Flow

Step	Description
	User fills out the input form (multi-step form UI).
	Frontend sends a POST request to /predict endpoint with JSON payload.
	Backend scales input data and runs ONNX model inference.
	Model returns probability → backend classifies into risk levels.
	Result sent back to frontend → shown in animated card with progress gauge.
	User can download report or retake the test.

5. Backend Highlights

✅ ONNX Inference Setup

```
const ort = require("onnxruntime-node");

const MODEL_PATH = path.join(__dirname, "models", "model.onnx");


async function loadModel() {
  session = await ort.InferenceSession.create(MODEL_PATH);
  console.log("🎯 Model loaded successfully!");
}
```

✅ Prediction Route

```
app.post("/predict", async (req, res) => {
  const scaled = manualScale(req.body);
  const input = new ort.Tensor("float32", Float32Array.from(scaled), [1, 13]);
  const result = await session.run({ float_input: input });

  const probs = Array.from(result.probabilities.data);
  const prob = probs[1];
  const risk = prob < 0.35 ? "Low" : prob < 0.65 ? "Moderate" : "High";

  res.json({ probability: +prob.toFixed(4), riskLevel: risk });
});
```

✅ CORS for Both Local & Vercel

```
const allowedOrigins = [
  "http://localhost:5173",
  "https://cvd-frontend.vercel.app"
];
app.use(cors({
  origin: (origin, cb) => !origin || allowedOrigins.includes(origin)
    ? cb(null, true)
    : cb(new Error("CORS not allowed"), false),
}));
```

💻 6. Frontend Highlights

✅ Smart API Selection

Automatically switches between local and deployed backend:

```
const API_BASE_URL =

 import.meta.env.VITE_API_URL ||

 (window.location.hostname === "localhost"

  ? "http://localhost:10000"

  : "https://cvd-backend-w8p6.onrender.com");
```

✅ Smooth Step UI

- Built with Framer Motion transitions
- 3-step flow: Personal → Clinical → Lifestyle
- Form validation and preset autofill (Low, Moderate, High Risk)

✅ Responsive Design

- Mobile-friendly Tailwind layout
- Footer non-overlapping
- Header fully transparent and adaptive
- Scalable buttons for touch devices

7. Features Summary

Feature	Description
🧠 AI Prediction	Predicts heart disease likelihood instantly
💬 Personalized Message	Custom risk messages based on result
📄 PDF Download	Generates detailed health report
🌙 Dark Mode	Toggle between light/dark themes
⚡ Fast API	ONNX runtime ensures low latency
🧍 User Presets	Quick example data for testing
📱 Mobile Responsive	Tailwind + adaptive layout

8. Deployment Process

🔹 Frontend (Vercel)

1. Push code to GitHub
2. Connect repository to Vercel
3. Set Build Command → npm run build
4. Output Directory → dist
5. Vercel auto-deploys on every push

🔹 Backend (Render)

1. Push backend to GitHub
2. Create new **Web Service** on Render
3. Set:
   - **Start Command** → node index.js
   - **Port** → process.env.PORT or 10000
4. Add models folder to repo

5. Render auto-detects build and deploys
6. Verified endpoint:
   https://cvd-backend-w8p6.onrender.com

9. Security & Stability

- CORS enabled for specific origins
- Express rate limiter → prevents abuse
- Proper JSON body validation
- Model loads once on startup → prevents memory leaks
- Error handling for cold starts

10. Outcome

✅ Fully functional **AI-based Heart Risk Prediction System**
✅ Optimized for **speed, security, and scalability**
✅ Works seamlessly in both **local development** and **cloud deployment**
✅ Modern, interactive, mobile-friendly UI/UX

Live URLs

Component	Platform	URL
Frontend (React)	**Vercel**	**🔗 https://cvd-frontend.vercel.app**
Backend (Node + ONNX)	**Render**	**🔗 https://cvd-backend-w8p6.onrender.com**

Conclusion

The **CVD Risk Predictor** successfully integrates **AI + Web Technologies** to deliver a fast, interactive, and accurate heart disease prediction system.
With modern deployment via **Vercel and Render**, it ensures **zero downtime**, **scalable inference**, and a **smooth UI** for users on any device.