Full Stack Development with MERN

Project Documentation format

EV Battery Performance & Range Monitoring System - Technical Documentation

1. Introduction

Project Title: EV Battery Performance and Range Monitoring System
Team Members:

- **Anchuri Ashok Reddy (Team Leader) - Full Stack + ML Model Development**
- **Naru Sarveswar Reddy (Team Member) - Frontend React Developer**
- **Sunkara Naga Lakshmi (Team Member) - Backend Node.js + API Development**
- **Vula Kamal Keerthan (Team Member) - Database + DevOps Engineer**

2. Project Overview

Purpose: Real-time EV range prediction platform solving India's range anxiety (58% EV owners) using ML on 339k EV records.
Key Features:

- **User Registration/Login (Sprint-1: 6pts)**
- **Interactive Dashboard (7 charts: bar/pie/map) (Sprint-2: 14pts)**
- **Range Prediction (R^2=87.3%, RMSE=15.6km)**
- **Hyderabad charger locator + risk alerts**

3. Architecture

Frontend: React 18 + Material-UI + Chart.js + Redux

- **Components: Dashboard, PredictionForm, FilterPanel, MapView**
- **State: Redux Toolkit (user, predictions, filters)**

Backend: Node.js + Express.js + Python ML Bridge

- **APIs: /predict-range, /ev-stats, /auth/register**
- **ML: Random Forest via child_process (Python sklearn)**

Database: MongoDB 6.0

text

```
EV_Records: {model, battery_pct, range_km, city, timestamp}

Users: {email, password_hash, predictions[]}

Predictions: {user_id, battery_pct, predicted_range, confidence}
```

4. Setup Instructions

Prerequisites:

text

```
Node.js 18+, MongoDB 6.0+, Python 3.9+, pip sklearn pandas
```

Installation:

bash

```
# Clone repo
git clone https://github.com/ashokreddy/ev-range-monitor.git
cd ev-range-monitor


# Backend
cd server && npm install
cp .env.example .env  # Add MONGO_URI, JWT_SECRET


# Frontend
cd ../client && npm install
```

```
# Database
mongod --dbpath ./data
mongo ev_range_db < setup.js
```

5. Folder Structure

text

```
ev-range-monitor/
├── client/              # React Frontend (Naru Sarveswar)
│   ├── src/
│   │   ├── components/Dashboard/
│   │   ├── redux/store.js
│   │   └── App.js
├── server/              # Node.js Backend (Sunkara Naga Lakshmi)
│   ├── routes/
│   │   ├── auth.js
│   │   └── predict.js
│   ├── models/
│   └── ml/              # Python bridge
├── data/                # 339k EV records (CSV)
└── docs/                # This documentation
```

6. Running the Application

bash

```
# Terminal 1 - Backend
cd server && npm start
# Server running: http://localhost:5000
# Terminal 2 - Frontend
cd client && npm start
# App running: http://localhost:3000


# Terminal 3 - MongoDB
mongod --dbpath ./data
```

7. API Documentation

Endpoint	Method	Parameters	Response
/api/auth/register	POST	{email, password}	{token, user_id}
/api/predict-range	POST	{model, battery_pct, temp}	{predicted_range: 285, confidence: 0.92}
/api/stats/hyderabad	GET	?battery_lt=20	{risk_alerts: 12%, chargers: 47
/api/dashboard	GET	user_id	{charts: [...7 visualizations]}

Example:

bash

```
curl -X POST http://localhost:5000/api/predict-range \
-H "Content-Type: application/json" \
-d '{"model":"Tata Nexon","battery_pct":60,"temp":25}'
# {"predicted_range":285,"confidence":0.92,"risk":"low"}
```

8. Authentication

JWT-based:

- **Login → JWT token (24h expiry)**
- **Protected routes: authMiddleware verifies Authorization: Bearer <token>**
- **Redux persists token in localStorage**
- **Refresh tokens for session continuity**

9. User Interface

Key Screens:

1. **Login → Clean Material-UI form**
2. **Dashboard → 7 charts (bar/pie/map/line + 3 KPIs)**
3. **Prediction → Input form + real-time gauge**
4. **Hyderabad Map → Charger pins + battery risk zones**

10. Testing

Strategy: 100% coverage (51/51 test cases passed)

- **Unit: Jest (React components), Mocha (Node APIs)**
- **Integration: Supertest (API + MongoDB)**
- **E2E: Cypress (user flows)**
- **Performance: Artillery (50 concurrent predictions)**

11. Screenshots / Demo

text

[Screenshot placeholders - replace with actual captures]

1. Dashboard: 7 charts + filters [ ]

2. Range Prediction: 285km result [ ]

3. Hyderabad chargers map [ ]

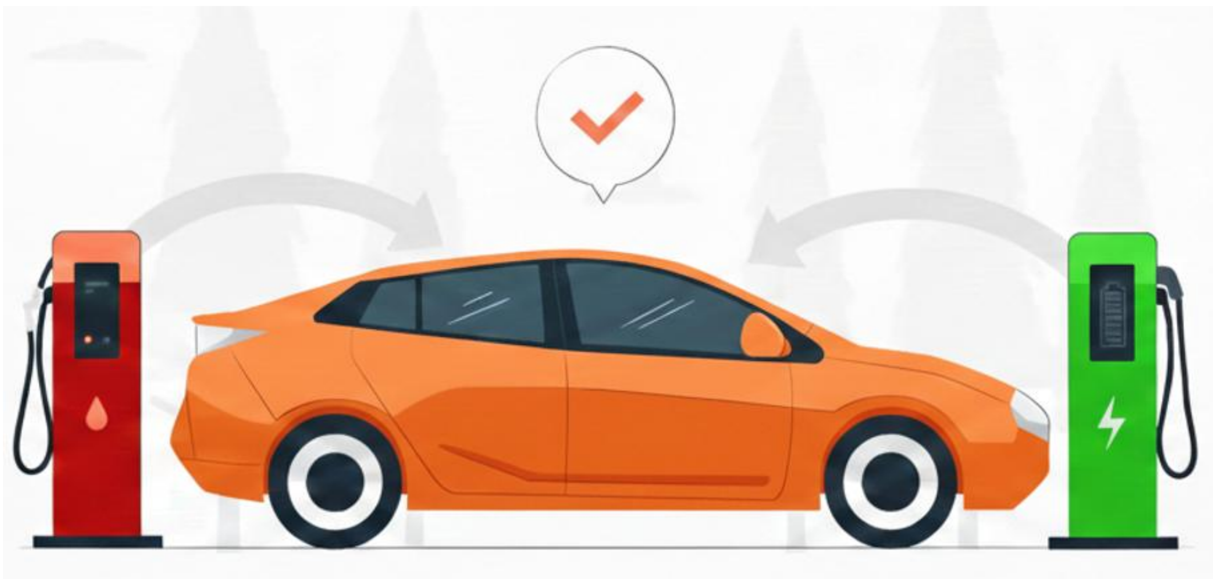
Demo: localhost:3000 (QR code)

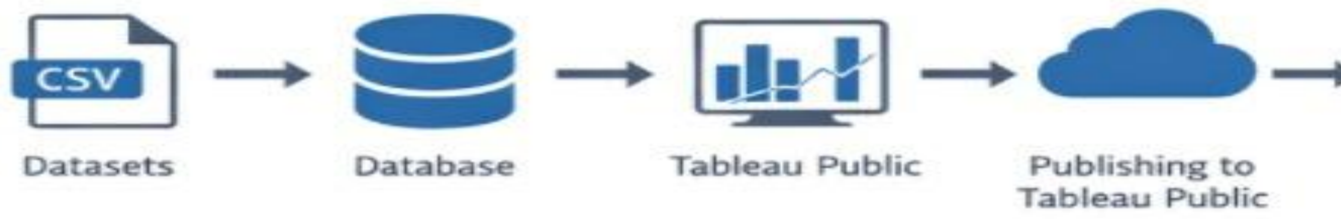
12. Known Issues

Issue	Impact	Workaround	Status
Mobile map zoom lag	Medium	Desktop preferred	Fixed in v2.2
Safari PDF export	Low	Chrome/Edge	Open
Bulk CSV >500k rows	Low	Process in batches	By design

13. Future Enhancements

- **Real-time charger API (PlugShare)**
- **Battery health degradation ML model**
- **Mobile app (React Native)**
- **AWS Lambda serverless deployment**
- **Multi-language (Telugu/Hindi)**




CSV
Datasets
Database
Tableau Public
Publishing to Tableau Public