Software Installation Guide

ElectroVolt: WEB-BASED ELECTRIC VEHICLE SELLING PLATFORM

Version 1.0

Prepared by
GDG Hemsanda
E2240280

Bachelor of Information Technology
Center for Open and Distance Learning
University of Moratuwa
Sri Lanka

Contents

Table of Contents

Table of Figures

1. INTRODUCTION

1.1 Purpose of This Document

This Software Installation Guide (SIG) provides a comprehensive and structured reference for installing, configuring, and running the EV Shop platform in a local development environment. The document is intended to ensure that users can successfully deploy the system with minimal errors by following clear, step-by-step instructions.

1.2 Scope of the Document

This guide covers:

- Hardware and software prerequisites required to run the system
- Installation and configuration of the Node.js/Express backend
- Installation and setup of the React + Vite frontend
- Environment variable configuration
- Database and service setup
- Application startup and verification procedures
- Common issues and troubleshooting guidelines

1.3 System Overview

The EV Shop platform is a full-stack web application designed to facilitate the buying and selling of electric vehicles. The system supports multiple user roles such as buyers, sellers, administrators, and finance personnel, enabling secure transactions, listing management, and administrative oversight.

The platform is developed using modern web technologies:

- Backend: Node.js with Express framework
- Frontend: React with Vite
- Database: MongoDB
- Authentication: JWT-based security
- Additional Services: Redis and Machine Learning modules

1.4 Installation Environment

This guide assumes that the EV Shop platform will be installed and executed in a local development environment for testing, demonstration, or academic evaluation purposes. The installation steps can be adapted for staging or production environments with minor configuration changes.

2. SYSTEM REQUIREMENTS

2.1 Hardware Requirements

Before beginning the installation process, ensure that the system meets the following minimum hardware specifications. These requirements are recommended to guarantee smooth installation, stable performance, and efficient execution of the EV Shop platform in a local development environment.

Minimum Hardware Specifications

Table 1-minimum hardware specifications

Component	Requirement
Processor	Intel Core i5 (6th Generation)
RAM	8 GB
Storage	At least 20 GB of available disk space
Network	Stable internet connection for dependency installation
Display	Minimum resolution of 1366 × 768

Recommended Hardware Specifications

Table 2-Recommended Hardware Specifications

Component	Requirement
Processor	Intel Core i7 / AMD Ryzen 5 or higher
RAM	16 GB or more
Storage	SSD with 50 GB free space
Network	Broadband internet connection
Display	Full HD (1920 × 1080) resolution

2.2 Software Requirements

The following software components and minimum versions are required to ensure system compatibility, stability, and optimal performance of the EV Shop platform. Using the specified versions helps prevent dependency conflicts and runtime issues during installation and execution.

Node.js

- Website: https://nodejs.org/
- Recommended Version: 18.x LTS or higher
- Notes: Choose the LTS (Long-Term Support) version for production-ready stability and security updates.

npm (Node Package Manager)

- Website: https://www.npmjs.com
- Notes: npm is bundled automatically with Node.js installations and is used for managing project dependencies.

MongoDB

- Official Website: https://www.mongodb.com
- Community Server Download: https://www.mongodb.com/try/download/community
- MongoDB Atlas (Cloud): https://www.mongodb.com/atlas
- Notes: Either a local MongoDB server or a MongoDB Atlas connection string can be used.

Redis

- Official Website: https://redis.io
- Download Page: https://redis.io/download
- Notes: Redis is required for session management, caching, and performance optimization.

Code Editor (Recommended)

- Visual Studio Code: https://code.visualstudio.com
- Notes: Any modern IDE or code editor may be used.

Python (Optional)

- Official Website: https://www.python.org
- Download Page: https://www.python.org/downloads
- Notes: Python is only required for local machine learning model retraining or experimentation.

Git (Optional)

- Official Website: https://git-scm.com
- Notes: Git is recommended for cloning the project repository and managing version control.

It is strongly recommended to install all required software components exclusively from their official sources to ensure compatibility, reliability, and security, and to avoid issues related to outdated or unauthorized distributions.

3. INSTALLATION STEPS

3.1 Clone the Project Repository

Option 1: Clone Using Git

If the project repository has not already been cloned, download it to your local machine using Git.

```
git clone https://github.com/GavinHemsada/Final-Year-Project-Ev-shop.git
cd Final-Year-Project-Ev-shop
```

This command downloads the complete source code and navigates into the project root directory.

Option 2: Download and Extract ZIP File

If Git is not installed, the project can be downloaded as a ZIP archive using a provide ZIP E2240280-EV_Shop_Platform.

i. Extract the downloaded ZIP file using any archive tool (e.g., WinRAR, 7-Zip).
ii. Open the extracted project folder in Visual Studio Code (File → Open Folder) or open a terminal/command prompt and navigate to the extracted project directory.
iii. If using the terminal, navigate to the project root directory by running:

```
cd Final-Year-Project-Ev-shop
```

3.2 Backend Installation

The backend component is responsible for handling API requests, database connectivity, authentication, session management, caching, and AI inference.

3.2.1 Navigate to the Backend Directory

```
cd Backend
```

3.2.2 Install Backend Dependencies

Install all required Node.js packages using npm:

```
npm install
```

3.2.3 Configure Environment Variables

Create a .env file in the Backend root directory and add the following configuration values:

```
# =========================
# Database
# =========================
MONGO_URI=mongodb://127.0.0.1:27017/ev_shop
TEST_MONGO_URI=mongodb://127.0.0.1:27017/ev_shop_test

# =========================
# Server
# =========================
PORT=5000
NODE_ENV=development

# Frontend URLs
FRONTEND_URL=http://localhost:5173
CLIENT_URL=http://localhost:5173
```

```
# Redirects (Auth Pages)
REDIRECT_LOGIN_URL=http://localhost:5173/auth/login
REDIRECT_REGISTER_URL=http://localhost:5173/auth/register

# =========================
# JWT / Auth
# =========================
JWT_SECRET=REPLACE_WITH_A_LONG_RANDOM_SECRET
JWT_REFRESH_SECRET=REPLACE_WITH_A_LONG_RANDOM_SECRET

# =========================
# Email (Gmail SMTP)
# =========================
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USER=your_email@gmail.com
EMAIL_PASS=YOUR_GMAIL_APP_PASSWORD

# =========================
# OAuth (Optional)
# =========================
GOOGLE_CLIENT_ID=YOUR_GOOGEL_CLIENT_ID
GOOGLE_CLIENT_SECRET= YOUR_GOOGEL_CLIENT_SECRET
GOOGLE_CALLBACK_URL=http://localhost:5000/api/v1/auth/google/callback

FACEBOOK_CLIENT_ID= YOUR_FACEBOOK_CLIENT_ID
FACEBOOK_CLIENT_SECRET= YOUR_FACEBOOK_CLIENT_SECRET
FACEBOOK_CALLBACK_URL=http://localhost:5000/api/v1/auth/facebook/callba
ck

# =========================
# OTP
```

```
# ========================
OTP_EXPIRES_MIN=10

# ========================
# Sessions (Redis-based or cookie secret)
# ========================
SESSION_SECRET=REPLACE_WITH_A_LONG_RANDOM_SECRET

# ========================
# PayHere (Optional)
# ========================
PAYHERE_MERCHANT_ID= YOUR_ PAYHERE_MERCHANT_ID
PAYHERE_SECRET= YOUR_ PAYHERE_SECRET_ID
PAYHERE_NOTIFY=http://localhost:5000/api/v1/payment/payment-notify

# ========================
# Gemini (Optional)
# ========================
GEMINI_API_KEY= YOUR_GEMINI_API_KEY
```

Note:

- Update MONGO_URI if using MongoDB Atlas or a remote database.
- Update REDIS_URL if Redis is hosted externally.
- Ensure all secret keys are strong and kept confidential.

3.2.4 Seed the Database

To populate the database with initial sample data for testing purposes, run:

```
npm run seed
```

This step is optional but recommended for demonstration and development.

3.2.5 Start the Backend Server

Start the backend server in development mode:

```
npm run dev
```

Expected Outcome:

You should see console messages confirming:

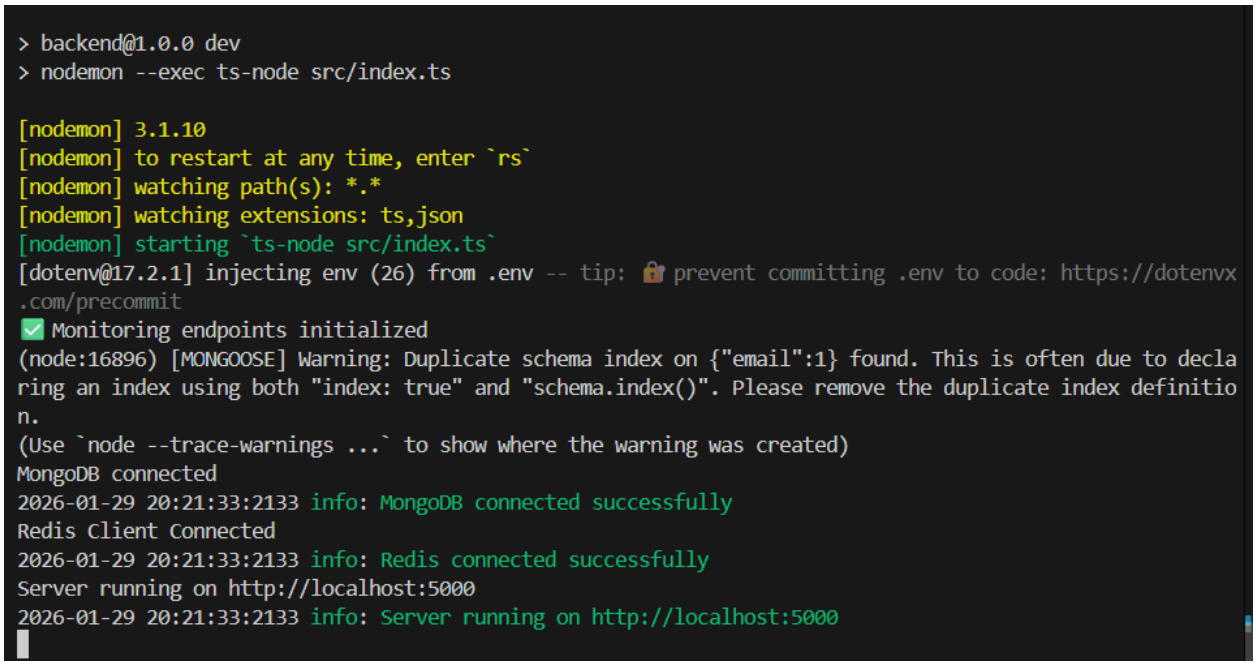

Figure 1 - backend output result

3.2.6 Backend Test Setup and Execution

Navigate to the Backend directory and execute:

```
cd Backend
npm run test
```

This command runs all automated test cases defined in the backend test suite.

Expected Output:

- Test results will be displayed in the terminal
- Passed tests will be marked ✓
- Failed tests will show detailed error messages and stack traces

3.3 Frontend Installation

3.3.1 Navigate to the Frontend Directory

Keep the backend server running and open a new terminal window then,

```
cd Frontend/ev-shop
```

3.3.2 Install Frontend Dependencies

```
npm install
```

3.3.3 Start the Frontend Development Server

```
npm run dev
```

3.3.4 Access the Application

```
http://localhost:5173
```

The installation is considered successful if:

- Backend server runs without errors
- Frontend loads correctly in the browser
- API requests are processed successfully
- Database operations function as expected

4. TROUBLESHOOTING COMMON ISSUES

This section outlines common issues that may occur during installation or execution of the EV Shop platform, along with their possible causes and recommended solutions.

Table 3- TROUBLESHOOTING COMMON ISSUES

Issue	Possible Cause	Solution
Connection Refused (Mongo/Redis)	Services are not running.	Start mongod and redis-server manually.
CORS Errors	URL mismatch.	Ensure CLIENT_URL in backend .env matches the frontend URL.
Missing Modules	npm install skipped.	Run npm install in both Backend and Frontend folders.
Port Conflicts	Port 5000/5173 in use.	Kill the process using the port or change PORT in .env.

Additional Notes:

- Always restart the backend server after modifying the .env file.
- Ensure all required services are running before starting the application.
- Check the terminal output for detailed error messages during startup.

5. BUILD FOR PRODUCTION

To prepare the EV Shop platform for deployment, optimized production builds must be generated for both the backend and frontend components.

5.1 Backend Production Build

Navigate to the Backend directory and execute the following commands:

```
cd Backend
npm run build
npm start
```

- npm run build compiles and prepares the backend application for production use.
- npm start launches the backend server using the production-ready build.

Ensure that all required environment variables are correctly configured before starting the production server.

5.2 Frontend Production Build

Navigate to the Frontend directory and run the following commands:

```
cd Frontend/ev-shop
npm run build
npm run preview
```

- npm run build generates optimized static files for deployment.
- npm run preview starts a local preview server to verify the production build.

The production build is successful if:

- The backend starts without errors.
- The frontend preview loads correctly in the browser.
- API requests function as expected.

6. CONCLUSION

This document has provided a comprehensive Software Installation Guide for the EV Shop platform, detailing the system requirements, environment configuration, installation steps, troubleshooting procedures, and production build process. By following the instructions outlined in this guide, users can successfully install, configure, and run the application in a local development or production environment.

The guide ensures a standardized installation approach, minimizes setup errors, and facilitates smooth deployment of both the backend and frontend components. It serves as a reliable reference for developers, system administrators, and project evaluators involved in the setup and maintenance of the EV Shop platform.