Intelligent Home Monitoring System Installation Guide

Click Here for Live Demo

Project Overview

The Intelligent Home Monitoring System (IHMS) is a full-stack capstone prototype for monitoring connected home devices, viewing power usage, tracking runtime, receiving alerts, and simulating emergency shutoff behavior.

The application is organized into three main parts:

- client/: React, TypeScript, Vite, Tailwind CSS frontend
- server/: Node.js and Express backend API
- database/: PostgreSQL setup script for recreating the demo database

The project is designed for local demonstration and Render deployment. The frontend includes fallback demo data so the interface can still be shown if the backend is temporarily unavailable.

Prerequisites

Install or create accounts for the following before setup:

- Node.js
- npm
- Git
- PostgreSQL
- Render account for deployment

Recommended versions:

- Node.js 20 or newer
- `npm 10 or newer`
- PostgreSQL 14 or newer

Clone the Repository

Open a terminal and run:

```
git clone https://github.com/eakillidev/ihms-demo
cd ihms
```

Install Backend Dependencies

From the root project directory:

```
cd server
npm install
```

This installs Express, PostgreSQL support, CORS middleware, and dotenv.

Install Frontend Dependencies

From the root project directory:

```
cd client
```

```
npm install
```

This installs React, Vite, Tailwind, Recharts, and the UI dependencies used by the frontend.

Environment File Setup

Environment files are required for local development, but they should not be committed to Git.

Backend Environment File

Create a .env file inside the server/ directory:

```
cd server
```

Create server/.env with the following example values:

```
DATABASE_URL=postgresql://username:password@host:port/database
PORT=5000
```

Replace the placeholder values with your local PostgreSQL database information.

Frontend Environment File

Create a .env file inside the client/ directory:

```
cd client
```

Create client/.env with:

```
VITE_API_BASE_URL=http://localhost:5000
```

This tells the frontend where to send API requests during local development.

Database Setup

Create a local PostgreSQL database named ihms if it does not already exist.

Example:

```
createdb ihms
```

From the root project directory, run the SQL setup file:

```
psql -U postgres -d ihms -f database/ihms_database.sql
```

This script drops and recreates the devices table, then inserts the demo device data used by the project.

Note: The backend also includes automatic database initialization for demo safety. The SQL file is provided so graders or developers can recreate the database state manually.

Start the Backend Locally

From the server/ directory:

```
npm start
```

The backend should start on:

```
http://localhost:5000
```

To confirm the backend is running, open:

```
http://localhost:5000
```

You should see a JSON response showing that the backend is working.

Test the API

Open this URL in a browser:

```
http://localhost:5000/api/devices
```

Or test from the terminal:

```
curl http://localhost:5000/api/devices
```

The response should contain an array of device records such as Living Room TV, Gaming Console, Kitchen Stove, Washer, Bedroom TV, and Office Monitor.

Start the Frontend Locally

Open a second terminal.

From the client/ directory:

```
npm run dev
```

Vite will print a local development URL, usually:

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

Open the URL in a browser.

After login or signup, the dashboard should display the demo devices and total power usage.

Common Troubleshooting

`npm install errors`

If dependency installation fails:

- Confirm Node.js and npm are installed.
- Run node -v and npm -v.
- Delete node_modules and package-lock.json only if dependency corruption is suspected, then rerun npm install.
- Make sure commands are being run inside the correct directory, either client/ or server/.

On Windows PowerShell, if npm scripts are blocked by execution policy, try:

```
npm.cmd install

npm.cmd run build
```

Database connection errors

If the backend cannot connect to PostgreSQL:

- Confirm PostgreSQL is running.
- Confirm the database exists.
- Check that server/.env contains DATABASE_URL.
- Confirm the username, password, host, port, and database name are correct.
- For Render, confirm the backend service has the DATABASE_URL environment variable set.

CORS or API connection errors

If the frontend loads but does not show backend data:

- Confirm the backend is running.
- `Open http://localhost:5000/api/devices directly.`
- `Confirm client/.env has VITE_API_BASE_URL=http://localhost:5000.`
- Restart the frontend after changing .env values.

Frontend not loading data

The frontend has fallback demo data, so the dashboard may still display devices even if the API is down.

To confirm whether backend data is loading:

- Open the browser developer console.
- Check for API warnings.
- Test the backend endpoint directly.
- Add a device and refresh the dashboard to confirm persistence.

Final Local Verification Checklist

Before submitting or presenting:

- Backend starts with npm start.
- `http://localhost:5000/api/devices returns device JSON.`
- Frontend starts with npm run dev.
- Dashboard loads successfully.
- Add Device flow creates a new device.
- Device modal opens and displays usage charts.
- Timer, alerts, and killswitch demo features still work.
- Frontend builds with npm run build.