Weather Dashboard

Overview

The Weather Dashboard is a comprehensive web application that displays real-time environmental data from a Raspberry Pi equipped with a SenseHAT sensor module, alongside external weather data. This dual-source approach allows users to compare local sensor readings with official weather forecasts, providing a rich and interactive weather monitoring experience.

Features

- Real-time display of SenseHAT sensor data (temperature, humidity, pressure)
- Integration with external weather API for current conditions and forecasts
- Interactive data visualization with charts and graphs
- Responsive design that works on desktop and mobile devices
- Dynamic background that changes based on current weather conditions

Architecture

The application follows a modern client-server architecture:

Frontend

- Single Page Application (SPA) built with React and Vite
- Component-based UI with responsive design
- Real-time data updates using React hooks and data fetching patterns
- Dynamic visualization using Chart.js

Backend

- Express.js server handling API requests
- MongoDB database for storing historical SenseHAT data
- RESTful API endpoints for data retrieval and submission
- Authentication middleware for securing SenseHAT data submission

Data Flow

1. SenseHAT sensors on Raspberry Pi collect environmental data
2. Data is sent to the Express backend via authenticated API endpoints
3. Frontend fetches both SenseHAT data and external weather data
4. Data is processed and displayed in the dashboard interface

Technology Stack

Frontend

- **React**: JavaScript library for building the user interface

- **Vite**: Next-generation frontend tooling for faster development
- **Chart.js**: JavaScript charting library for data visualization
- **TailwindCSS**: Utility-first CSS framework for styling
- **Axios**: Promise-based HTTP client for API requests

Backend

- **Node.js**: JavaScript runtime for the server
- **Express**: Web application framework for Node.js
- **MongoDB**: NoSQL database for storing sensor data
- **Mongoose**: MongoDB object modeling for Node.js
- **dotenv**: Module for loading environment variables

DevOps

- **Docker**: Containerization platform for deployment
- **Nginx**: Web server for serving the production build
- **Docker Compose**: Tool for defining multi-container Docker applications

Local Development Setup

Prerequisites

- Node.js (v14.x or higher)
- npm (v6.x or higher)
- MongoDB (local or remote instance)

Frontend Setup

1. Clone the repository:

```
git clone https://github.com/yourusername/weather-dashboard.git
cd weather-dashboard
```

2. Install dependencies:

```
npm install
```

3. Create a `.env` file in the project root with the following content:

```
VITE_WEATHER_API_KEY=your_weather_api_key
VITE_API_BASE_URL=http://localhost:5000/api
```

4. Start the development server:

```
npm run dev
```

5. Open your browser and navigate to `http://localhost:5173`

Backend Setup

1. Navigate to the backend directory:

```
cd backend
```

2. Install backend dependencies:

```
npm install
```

3. Create a `.env` file in the backend directory with the following content:

```
PORT=5000
MONGODB_URI=your_mongodb_uri
```

4. Start the backend server:

```
npm run dev
```

Connecting to Raspberry Pi SenseHAT

Raspberry Pi Setup

1. Install the required libraries on your Raspberry Pi:

```
sudo apt-get update
sudo apt-get install -y python3-pip
pip3 install requests sense-hat
```

2. Create a Python script (e.g., `send_sensehat_data.py`) to read SenseHAT data and send it to your backend:

```
import time
import requests
```

```
from sense_hat import SenseHat

sense = SenseHat()
API_ENDPOINT = "http://your-server-ip:5000/api/sensehat/data"
API_KEY = "your_api_key_from_backend_env"

while True:
    temperature = sense.get_temperature()
    humidity = sense.get_humidity()
    pressure = sense.get_pressure()

    data = {
        "temperature": temperature,
        "humidity": humidity,
        "pressure": pressure
    }

    headers = {
        "x-api-key": API_KEY,
        "Content-Type": "application/json"
    }

    try:
        response = requests.post(API_ENDPOINT, json=data,
headers=headers)
        print(f"Data sent. Response: {response.status_code}")
    except Exception as e:
        print(f"Error sending data: {str(e)}")

    time.sleep(60)  # Send data every minute
```

3. Run the script on your Raspberry Pi:

```
python3 send_sensehat_data.py
```

Configuring the Dashboard for Local SenseHAT Data

1. Make sure your Raspberry Pi and the machine running the Weather Dashboard are on the same network.
2. Update the `.env` file in your backend directory with the correct API key:

```
API_KEY=same_key_as_in_raspberry_pi_script
```

3. If running the application with Docker, update the backend service in `docker-compose.yml` to expose the API endpoint to your local network:

```
backend:
  # other configs...
  environment:
    - API_KEY=your_api_key
  ports:
    - "5000:5000"
```

4. Restart the application to apply the changes:

```
docker compose down
docker compose up -d
```

Troubleshooting

Common Issues

1. **SenseHAT data not appearing in the dashboard**
   - Check that the Raspberry Pi script is running and sending data
   - Verify the API key matches between the Raspberry Pi script and backend
   - Check network connectivity between the Raspberry Pi and backend server
2. **Weather background video not loading in Docker**
   - Ensure that assets are properly imported in the React components
   - Verify that the Nginx configuration is correctly serving static files
3. **Backend API connection failing**
   - Check that the backend server is running and accessible
   - Verify that the correct API base URL is set in the frontend `.env` file

Contributing

Contributions are welcome! Please feel free to submit a Pull Request.

License

This project is licensed under the MIT License - see the LICENSE file for details.

Weather Application Deployment Guide

This guide provides detailed instructions for deploying the Weather application using Docker Compose on your VPS.

Prerequisites

- Docker and Docker Compose installed on your VPS
- Domain name (weather.emmi.zone) configured to point to your VPS IP
- Basic understanding of terminal commands and Docker

Project Overview

This is a React-based weather application built with Vite that displays weather data, forecasts, and visualizations. The application requires:

- OpenWeather API key for weather data
- Proper environment configuration for production deployment
- Port 5137 exposed for web access

Deployment Instructions

1. Clone the Repository

```
git clone <repository-url>
cd weather-app
```

2. Environment Variables Configuration

The application requires specific environment variables for proper functionality. Create a `.env.production` file in the project root with the following variables:

```
VITE_WEATHER_API_KEY=your_openweather_api_key
VITE_SENSEHAT_API_URL=https://api.emmi.zone/sensehat
```

Replace `your_openweather_api_key` with your actual OpenWeather API key.

Important Notes about Environment Variables:

- The `.env` file is ignored in Docker builds for security reasons (specified in `.dockerignore`)
- Environment variables are injected at runtime via the Docker Compose configuration
- For production, values are read from `.env.production` or can be passed directly to the docker-compose command

3. Docker Compose Configuration

The `docker-compose.yml` file is configured to:

- Build the application using the multi-stage Dockerfile
- Expose the application on port 5137
- Set the required environment variables
- Configure Nginx to serve the application

- Enable automatic container restarts

4. Building and Starting the Application

To build and start the application:

```
# Build and start in detached mode
docker-compose up -d --build

# View the logs to check for any issues
docker-compose logs -f
```

During the build process, Docker will:

1. Install all NPM dependencies
2. Build the React application with Vite
3. Configure Nginx to serve the static files
4. Expose the application on port 5137

5. Accessing the Application

Once deployed, the application will be accessible at:

```
http://weather.emmi.zone:5137
```

If you've configured your domain to handle the port internally, you may be able to access it via:

```
http://weather.emmi.zone
```

6. NGINX Configuration

The application uses Nginx to serve the static files. The default configuration in `nginx.conf` includes:

- Server configuration for the domain `weather.emmi.zone`
- Static file serving for the React application
- API proxying to handle backend requests
- Error page handling

7. Managing the Deployment

Updating the Application

To update the application with the latest code:

```
git pull
docker-compose down
docker-compose up -d --build
```

Stopping the Application

```
docker-compose down
```

Viewing Logs

```
docker-compose logs -f
```

Troubleshooting

Common Issues

1. **Application not accessible**
   - Check that port 5137 is open in your firewall
   - Verify that Docker containers are running with `docker-compose ps`
   - Check logs for errors with `docker-compose logs -f`
2. **Environment variable issues**
   - Verify that `.env.production` exists and contains the correct values
   - Check if environment variables are correctly passed to Docker Compose
3. **Build failures**
   - Ensure all dependencies are correctly specified in package.json
   - Check disk space and memory available on your VPS
4. **Nginx configuration issues**
   - Verify that the nginx.conf file is correctly mounted in the container
   - Check Nginx logs for configuration errors

Security Considerations

- The OpenWeather API key is included in the frontend build. While this is generally acceptable for weather APIs, consider using a backend service to hide API keys for more sensitive services.
- Consider setting up HTTPS for production use to encrypt data transmission.
- Regularly update dependencies to address security vulnerabilities.

Backup and Recovery

It's recommended to:

- Backup your .env.production file
- Maintain a copy of your docker-compose.yml and Dockerfile
- Document any custom configurations specific to your deployment

Additional Resources

- Docker Documentation
- Docker Compose Documentation
- Nginx Documentation
- Vite Documentation

Weather Dashboard with SenseHAT Integration

Project Overview

This Weather Dashboard is a sophisticated web application that combines real-world weather data with local SenseHAT sensor readings. The project demonstrates the integration of multiple data sources, real-time updates, and interactive visualization capabilities, all while maintaining a modern, responsive user interface.

Purpose

The primary goal of this project is to create an interactive platform that:

1. Fetches and displays current weather data from OpenWeatherMap API
2. Integrates with a locally hosted SenseHAT device for environmental readings
3. Provides a comparative analysis between official weather data and local sensor readings
4. Visualizes weather patterns through an interactive map interface
5. Delivers weather insights through an automated cycling system

Technical Stack

Frontend Technologies

- **React 18.2.0**: Core framework for building the user interface
- **Vite 5.1.0**: Next-generation frontend tooling
- **TailwindCSS 3.4.1**: Utility-first CSS framework
- **Framer Motion 12.6.2**: Animation library
- **Leaflet 1.9.4**: Interactive mapping
- **Chart.js & React-Chartjs-2**: Data visualization
- **Axios 1.6.7**: HTTP client
- **Radix UI**: Accessible component primitives

Backend Technologies

- **Node.js & Express 4.18.2**: Server framework
- **MongoDB & Mongoose 7.0.3**: Database and ODM
- **CORS**: Cross-origin resource sharing
- **dotenv**: Environment configuration

APIs and External Services

- **OpenWeatherMap API**: Weather data and forecasting
- **OpenStreetMap**: Base map layer for weather visualization
- **GeoCoding API**: Location search and coordinates conversion
- **SenseHAT API**: Raspberry Pi sensor data

Project Structure

```
weather-dashboard/
├── src/                          # Frontend source code
│   ├── components/              # React components
│   ├── hooks/                   # Custom React hooks
│   └── utils/                   # Utility functions
├── backend/                      # Node.js backend
│   ├── config/                  # Configuration files
│   ├── middleware/              # Express middleware
│   ├── models/                  # Mongoose models
│   ├── routes/                  # API routes
│   └── server.js                # Server entry point
├── public/                       # Static assets
├── dist/                         # Production build
├── package.json                 # Frontend dependencies
├── vite.config.js               # Vite configuration
├── tailwind.config.js           # Tailwind configuration
└── .env                         # Environment variables
```

Setup and Installation

Prerequisites

- Node.js (v16 or higher)
- MongoDB
- Raspberry Pi with SenseHAT module
- OpenWeatherMap API key

Frontend Setup

1. Clone the repository

```
git clone <repository-url>
cd weather-dashboard
```

2. Install frontend dependencies

```
npm install
```

3. Create a `.env` file in the root directory

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

4. Start the development server

```
npm run dev
```

5. Open your browser and navigate to `http://localhost:5173`

Mock Data Configuration

By default, the application uses mock data to simulate SenseHAT readings for development purposes. This is controlled by the `USE_MOCK_DATA` flag in `src/hooks/useSenseHatData.js`.

To switch between mock and real SenseHAT data:

1. Open `src/hooks/useSenseHatData.js`
2. Locate the `USE_MOCK_DATA` constant near the top of the file:

```
const USE_MOCK_DATA = true; // Set to false to use real API
```

3. Set it to `false` to use real SenseHAT data from your backend API
4. Set it to `true` to use mock data for development

Note: When using real data, ensure your backend server is running and properly configured to receive data from your Raspberry Pi SenseHAT.

Backend Setup

1. Navigate to the backend directory

```
cd backend
```

2. Install backend dependencies

```
npm install
```

3. Create a `.env` file in the backend directory

```
PORT=5000
MONGODB_URI=your_mongodb_uri
```

4. Start the backend server

```
npm run dev
```

Connecting to SenseHAT

Hardware Setup

1. Ensure SenseHAT is properly attached to your Raspberry Pi
2. Connect Raspberry Pi to your local network
3. Note down the Raspberry Pi's IP address

Software Configuration

1. The backend server communicates with the SenseHAT through a dedicated API
2. Environment variables control the connection to the SenseHAT device
3. Real-time data updates are handled through WebSocket connections

Development Features

Frontend

- Modern React with hooks and functional components
- Responsive design with Tailwind CSS
- Interactive maps with Leaflet.js
- Real-time data visualization with Chart.js
- Smooth animations with Framer Motion
- Accessible UI components with Radix UI

Backend

- RESTful API design
- MongoDB database integration
- Environment-based configuration
- CORS support for cross-origin requests
- Error handling middleware

- API rate limiting

Performance Optimization

- Vite for fast development and optimized builds
- Lazy loading of components
- Debounced API calls
- Efficient state management
- Optimized animations

Error Handling

- Graceful fallbacks for API failures
- User-friendly error messages
- Connection status indicators
- Automatic retry mechanisms

Future Enhancements

- Historical data visualization
- Advanced weather predictions
- Additional sensor integrations
- Data export capabilities
- Custom alert systems

Author

Emmi
emmi.zone

License