Giant Sloth Orchard Weather Server - Development Setup Guide

🌺 Local Development Environment Setup

Prerequisites

- Node.js 14.0.0 or higher
- npm 6.0.0 or higher
- Git (optional, for version control)

Quick Start (5 Minutes)

1. **Download and Setup the Proxy Server**

   bash

   ```bash
   # Create project directory
   mkdir giant-sloth-weather-server
   cd giant-sloth-weather-server

   # Save the proxy-server.js and package.json files to this directory
   # Then install dependencies
   npm install
   ```

2. **Create the Public Directory Structure**

   bash

   ```bash
   mkdir public
   # Copy your website files (index.html, styles.css, *.js) to the public/ folder
   ```

3. **Start the Development Server**

   bash

   ```bash
   npm run dev
   # or for simple start:
   npm start
   ```

4. **Access Your Weather Station**
   - Open browser to: `http://localhost:3000`
   - Your Giant Sloth Orchard site will load with full weather functionality

Directory Structure

```
giant-sloth-weather-server/
├── proxy-server.js        # Main server file
├── package.json           # Dependencies and scripts
├── public/                # Your website files
│   ├── index.html
│   ├── styles.css
│   ├── plant-manager.js
│   ├── weather-manager.js
│   ├── map-manager.js
│   ├── app.js
│   ├── giantslothorchard.png
│   ├── giantslothorchard_map.png
│   └── plantdatabase.js
└── logs/                  # Auto-created for logs
```

🔧 Weather Configuration in Development

1. Demo Mode (Default - Works Immediately)

- **Setup**: No configuration needed
- **Features**: Realistic weather patterns, daily cycles, random variations
- **Status**: Green LED, shows "demo" in footer
- **Usage**: Perfect for testing and development

2. Cloud API Mode (WeatherLink v2 Cloud)

1. **Get Credentials** from Davis Instruments WeatherLink.com:
   - API Key
   - Station ID
   - API Secret
2. **Configure in Weather Settings**:
   - Click Weather page → Settings gear (⚙️)
   - Enter Cloud API credentials
   - Click "Test Cloud API" to verify
   - Switch to "Cloud Mode"
3. **Proxy URL Configuration**:

```javascript
// In weather-manager.js, the cloud API will use:
// Local proxy: http://localhost:3000/api/cloud/current/{stationId}
// This automatically handles CORS and API signatures
```

3. Local API Mode (Direct WeatherLink Device)

1. **Find Your Device IP**:
   - Use network discovery: `http://localhost:3000/api/discover`
   - Or check your router's connected devices
   - Default WeatherLink Live IP: `192.168.1.100`
2. **Configure in Weather Settings**:
   - Enter device IP address (e.g., `192.168.1.100`)
   - Set port (usually `80`)
   - Enable HTTPS if your device uses it
   - Set proxy URL: `http://localhost:3000`
   - Click "Test Local API" to verify
   - Switch to "Local Mode"

4. Auto Mode

- **Setup**: Configure both Cloud and Local API credentials
- **Behavior**: Tries Cloud → Local → Demo (in that order)
- **Status**: Shows actual working mode in footer

🚀 Development Server Features

Available Endpoints

```bash
# Your weather station dashboard
http://localhost:3000/

# API endpoints for testing
http://localhost:3000/api/status          # Server status
http://localhost:3000/api/health          # Health check
http://localhost:3000/api/demo/current    # Demo weather data
http://localhost:3000/api/discover        # Find WeatherLink devices
```

Testing API Connections

bash

```bash
# Test cloud API
curl -X POST http://localhost:3000/api/test/cloud \
  -H "Content-Type: application/json" \
  -d '{"apiKey":"your-key","stationId":"your-id","apiSecret":"your-secret"}'

# Test local API
curl -X POST http://localhost:3000/api/test/local \
  -H "Content-Type: application/json" \
  -d '{"ip":"192.168.1.100","port":"80","https":false}'
```

Development Scripts

bash

```bash
npm run dev       # Start with auto-restart (recommended)
npm start         # Start server normally
npm run health    # Check server health
npm run status    # Get detailed status
npm run demo      # View demo data
npm run discover  # Find WeatherLink devices on network
```

🔍 Troubleshooting Development Issues

Common Problems and Solutions

1. **"Cannot connect to proxy server" Error**

   bash

   ```bash
   # Check if server is running
   curl http://localhost:3000/api/health

   # Restart the server
   npm run dev
   ```

2. **Local API Connection Fails**

   bash

   ```bash
   # Test device directly
   curl http://192.168.1.100/v1/current_conditions

   # Use discovery to find devices
   curl http://localhost:3000/api/discover
   ```

3. **Cloud API Authentication Errors**

- Verify credentials at WeatherLink.com
- Check API quota/limits
- Ensure Station ID is correct

4. **CORS Errors in Browser**
   - Make sure you're accessing via `http://localhost:3000` (not file://)
   - Server handles all CORS automatically

Debug Mode

```bash
# Enable verbose logging
DEBUG=* npm run dev

# View server logs
npm run logs
```

📊 Testing Weather Functionality

Complete Test Sequence

1. **Start Development Server**: `npm run dev`
2. **Open Dashboard**: `http://localhost:3000`
3. **Test Demo Mode**: Should work immediately with green LED
4. **Test Local API**:
   - Enter local device IP in settings
   - Click "Test Local API"
   - Switch to Local Mode if successful
5. **Test Cloud API**:
   - Enter WeatherLink credentials in settings
   - Click "Test Cloud API"
   - Switch to Cloud Mode if successful
6. **Test Auto Mode**: Should intelligently switch between available APIs

Expected Behavior

- **LED Colors**: 🔴 Red (disconnected) → 🟡 Yellow (connecting) → 🟢 Green (connected)
- **Footer Status**: Shows actual active mode (`demo`/`cloud`/`local`/`connecting`/`disconnected`)
- **Data Updates**: Live updates every 1 second (configurable in settings)

- **Charts**: Historical data visualization with 24h/7d/30d ranges

🎯 Development Best Practices

File Organization

- Keep website files in `public/` directory
- Server files in root directory
- Use `npm run dev` for development (auto-restart)
- Check `http://localhost:3000/api/status` for server health

Performance Tips

- Demo mode is fastest for UI development
- Local API mode is best for real device testing
- Cloud API mode for production-like testing
- Use browser dev tools to monitor network requests

Security Notes

- Proxy server handles all API authentication
- No credentials stored in browser
- CORS properly configured for development
- All API calls proxied through localhost:3000

✅ Success Indicators

Your development setup is working correctly when:

1. ✅ Server starts without errors: `npm run dev`
2. ✅ Dashboard loads: `http://localhost:3000`
3. ✅ Demo mode shows green LED and live weather data
4. ✅ Settings panel opens and saves configurations
5. ✅ API tests return success (if credentials configured)
6. ✅ Weather widgets update in real-time
7. ✅ Charts display data with proper tropical styling

🎉 Next Steps

Once development setup is working:

1. Test all weather modes with your actual WeatherLink device

2. Customize plant database and map layout
3. Configure update intervals for your needs
4. Export/backup weather data for testing
5. Prepare for production deployment

🦥 **Happy developing with Giant Sloth Orchard Weather Station!**