🍎 Mac Local Deployment Guide - SOaC Framework

Complete guide for running the SOaC Framework Phase 2 on macOS for development and testing.

📋 Table of Contents

🎯 Prerequisites

Required Software

- **macOS**: 10.15 (Catalina) or later
- **Homebrew**: Package manager for macOS
- **Node.js**: Version 18 or later
- **npm**: Comes with Node.js
- **Git**: For version control

Recommended Software

- **Visual Studio Code**: Code editor with excellent Node.js support
- **iTerm2**: Better terminal alternative
- **Postman**: For API testing

💻 System Requirements

Component	Minimum	Recommended
macOS	10.15+	12.0+
RAM	4 GB	8 GB+
Disk Space	2 GB	5 GB+
Node.js	18.x	20.x+
Internet	Required for installation	-

🔧 Installation Methods

Choose one of the following methods:

Method 1: Automated Setup (Recommended)

✅ Quick and easy
✅ Checks prerequisites automatically
✅ Installs all dependencies
⏱️ Time: 10-15 minutes

Method 2: Manual Setup

✅ Full control over each step
✅ Better for troubleshooting
✅ Educational
⏱️ Time: 20-30 minutes

🛠️ Manual Setup

Step 1: Install Homebrew

If you don’t have Homebrew installed:

```
# Install Homebrew
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/in-
stall.sh)"

# Verify installation
brew --version
```

Step 2: Install Node.js and npm

```
# Install Node.js (includes npm)
brew install node@20

# Or install LTS version
brew install node

# Verify installation
node --version   # Should show v18.x.x or higher
npm --version    # Should show 9.x.x or higher
```

Alternative: Using nvm (Node Version Manager)

```
# Install nvm
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh | bash

# Reload shell configuration
source ~/.zshrc  # or source ~/.bash_profile

# Install Node.js LTS
nvm install --lts

# Use installed version
nvm use --lts

# Set default
nvm alias default node
```

Step 3: Clone or Navigate to Project

If you already have the project:

```
cd /path/to/soac-framework
```

If cloning from GitHub:

```
# Navigate to your projects directory
cd ~/Projects  # or your preferred location

# Clone repository
git clone https://github.com/ge0mant1s/soac-framework.git

# Navigate into project
cd soac-framework
```

Step 4: Install Backend Dependencies

```
# Navigate to backend directory
cd backend

# Install dependencies
npm install

# This installs:
# - express (web framework)
# - cors (Cross-Origin Resource Sharing)
# - dotenv (environment variables)
# - bcryptjs (password hashing)
# - jsonwebtoken (JWT authentication)
# - morgan (HTTP logging)
# - helmet (security headers)
# - express-rate-limit (rate limiting)
```

Step 5: Configure Backend Environment

```
# Create .env file from example
cp .env.example .env

# Edit .env file
nano .env  # or use your preferred editor
```

Update the following values in `.env`:

```
# Server Configuration
PORT=5000
NODE_ENV=development

# JWT Configuration
JWT_SECRET=your_unique_secret_key_change_this_NOW
JWT_EXPIRES_IN=24h

# Admin Credentials
ADMIN_USERNAME=admin
ADMIN_PASSWORD=SecurePassword123!  # Change this!

# API Rate Limiting
RATE_LIMIT_WINDOW_MS=900000
RATE_LIMIT_MAX_REQUESTS=100

# CORS Configuration
CORS_ORIGIN=http://localhost:3000

# Optional: Palo Alto NGFW (for future use)
PALOALTO_API_URL=https://your-firewall.local/api
PALOALTO_API_KEY=your_api_key

# Optional: EntraID (for future use)
ENTRAID_TENANT_ID=your_tenant_id
ENTRAID_CLIENT_ID=your_client_id
ENTRAID_CLIENT_SECRET=your_client_secret
```

⚠️ **Security Note**: Always change `JWT_SECRET` and admin credentials before deployment!

Step 6: Install Frontend Dependencies

```
# Navigate to frontend directory
cd ../frontend

# Install dependencies
npm install

# This installs:
# - react (UI library)
# - react-dom (React DOM renderer)
# - react-router-dom (routing)
# - axios (HTTP client)
# - @mui/material (Material-UI components)
# - @mui/icons-material (Material-UI icons)
# - recharts (charts and graphs)
# - vite (build tool and dev server)
```

Step 7: Configure Frontend Environment

```
# Create .env file from example
cp .env.example .env

# Edit .env file
nano .env  # or use your preferred editor
```

Update the following values in `.env`:

```
# API Configuration
VITE_API_BASE_URL=http://localhost:5000/api

# Application Configuration
VITE_APP_NAME=SOaC Framework
VITE_APP_VERSION=1.0.0
```

Step 8: Verify Installation

```
# Check backend dependencies
cd ../backend
npm list --depth=0

# Check frontend dependencies
cd ../frontend
npm list --depth=0

# Both should show all packages installed successfully
```

🤖 Automated Setup

Use the provided setup script for quick installation:

Step 1: Make Script Executable

```
cd soac-framework
chmod +x scripts/setup-mac.sh
```

Step 2: Run Setup Script

```
./scripts/setup-mac.sh
```

The script will:
1. ✅ Check for Homebrew (install if missing)
2. ✅ Check for Node.js (install if missing)
3. ✅ Install backend dependencies
4. ✅ Create backend .env file
5. ✅ Install frontend dependencies
6. ✅ Create frontend .env file
7. ✅ Verify installation

Step 3: Configure Environment Variables

After the script completes, edit the environment files:

```
# Edit backend environment
nano backend/.env

# Edit frontend environment
nano frontend/.env
```

🚀 Running the Application

Method 1: Two Terminal Windows (Recommended)

Terminal 1 - Backend:

```
cd soac-framework/backend
npm run dev
```

You should see:

```
[nodemon] starting `node src/server.js`
✓ Server running on http://localhost:5000
✓ Environment: development
✓ CORS enabled for: http://localhost:3000
```

Terminal 2 - Frontend:

```
cd soac-framework/frontend
npm run dev
```

You should see:

```
  VITE v5.0.8  ready in 1234 ms

  ➜  Local:   http://localhost:3000/
  ➜  Network: use --host to expose
  ➜  press h to show help
```

Method 2: Background Processes

Start backend in background:

```
cd backend
npm run dev &> backend.log &
echo $! > backend.pid
```

Start frontend in background:

```
cd frontend
npm run dev &> frontend.log &
echo $! > frontend.pid
```

Stop processes:

```
# Stop backend
kill $(cat backend/backend.pid)

# Stop frontend
kill $(cat frontend/frontend.pid)
```

Method 3: Using PM2 Process Manager

Install PM2:

```
npm install -g pm2
```

Start both services:

```
# Start backend
cd backend
pm2 start npm --name "soac-backend" -- run dev

# Start frontend
cd ../frontend
pm2 start npm --name "soac-frontend" -- run dev

# View status
pm2 status

# View logs
pm2 logs

# Stop all
pm2 stop all

# Restart all
pm2 restart all
```

🌐 Accessing the Application

Frontend (React UI)

Open your browser and navigate to:

```
http://localhost:3000
```

Backend API

API endpoints available at:

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

Health Check

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

Should return:

```
{
  "status": "healthy",
  "timestamp": "2024-11-13T12:00:00.000Z",
  "environment": "development"
}
```

🧪 Testing

Test Backend API

1. Health Check:

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

2. Login (Get JWT Token):

```
curl -X POST http://localhost:5000/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "username": "admin",
    "password": "admin123"
  }'
```

Response:

```
{
  "success": true,
  "data": {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "user": {
      "id": 1,
      "username": "admin",
      "role": "admin"
    }
  }
}
```

3. Test Protected Route (Get Devices):

```
# Replace YOUR_TOKEN with the token from login response
curl http://localhost:5000/api/devices \
  -H "Authorization: Bearer YOUR_TOKEN"
```

Test Frontend

1. **Open Browser**: Navigate to http://localhost:3000
2. **Login Page**: Should see SOaC Framework login page
3. **Login**: Use admin/admin123 (or your configured credentials)
4. **Dashboard**: Should redirect to dashboard after successful login
5. **Navigation**: Test all menu items:
   - Dashboard
   - Devices
   - Rules
   - Alerts
   - Palo Alto Config
6. **Logout**: Test logout functionality

Run Automated Tests (if available)

```
# Backend tests
cd backend
npm test

# Frontend tests
cd frontend
npm test
```

🔧 Troubleshooting

Issue: "Port 5000 already in use"

Solution 1: Kill process on port 5000

```
# Find process
lsof -ti:5000

# Kill process
lsof -ti:5000 | xargs kill -9
```

Solution 2: Change port

```
# Edit backend/.env
PORT=5001

# Update frontend/.env to match
VITE_API_BASE_URL=http://localhost:5001/api
```

Issue: "Port 3000 already in use"

Solution 1: Kill process

```
lsof -ti:3000 | xargs kill -9
```

Solution 2: Use different port

```
# Vite will automatically suggest port 3001
# Or manually specify in frontend/vite.config.js
```

Issue: "Cannot find module"

Solution: Reinstall dependencies

```
# Backend
cd backend
rm -rf node_modules package-lock.json
npm install

# Frontend
cd frontend
rm -rf node_modules package-lock.json
npm install
```

Issue: “CORS error in browser console”

Error: `Access to XMLHttpRequest at 'http://localhost:5000/api/...' from origin 'http://localhost:3000' has been blocked by CORS policy`

Solution: Check backend CORS configuration

```
# Verify CORS_ORIGIN in backend/.env
CORS_ORIGIN=http://localhost:3000

# Restart backend
cd backend
npm run dev
```

Issue: “JWT token invalid”

Solution: Verify JWT_SECRET matches

```
# Check JWT_SECRET in backend/.env
# Ensure it's not empty and is properly set

# Clear browser localStorage
# Open browser console (F12) and run:
localStorage.clear()

# Then login again
```

Issue: “npm install fails”

Solution 1: Clear npm cache

```
npm cache clean --force
npm install
```

Solution 2: Use different registry

```
npm install --registry https://registry.npmjs.org/
```

Solution 3: Check Node.js version

```
node --version
# Ensure it's 18.x or higher

# If not, update Node.js
brew upgrade node
```

Issue: “Command not found: brew”

Solution: Install Homebrew

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/in-
stall.sh)"
```

Issue: “Permission denied”

Solution: Fix permissions

```
# For npm global packages
sudo chown -R $USER /usr/local/lib/node_modules

# For project directory
chmod -R 755 soac-framework/
```

Issue: “Backend connects but frontend shows blank page”

Solution: Check browser console

1. Open browser console (F12 → Console tab)
2. Look for error messages
3. Common issues:
   - API URL incorrect in frontend/.env
   - Backend not running
   - CORS not configured

Issue: “Frontend displays but can’t login”

Solution: Verify backend API

```
# Test login endpoint
curl -X POST http://localhost:5000/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"admin123"}'

# Should return token and user data
```

🎯 Development Tips

Hot Reload

Both backend and frontend support hot reload:
- **Backend**: nodemon watches for file changes
- **Frontend**: Vite watches for file changes

Simply save your files and see changes immediately!

Debugging

Backend Debugging:

```bash
# Enable debug logging
NODE_ENV=development npm run dev

# Use Node.js debugger
node --inspect src/server.js
```

Frontend Debugging:

```bash
# Use React Developer Tools (browser extension)
# Or open browser console (F12)
```

VS Code Configuration

Create `.vscode/launch.json` for debugging:

```json
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "node",
      "request": "launch",
      "name": "Debug Backend",
      "program": "${workspaceFolder}/backend/src/server.js",
      "envFile": "${workspaceFolder}/backend/.env"
    }
  ]
}
```

Useful Commands

```bash
# View backend logs
tail -f backend/backend.log

# View frontend logs
tail -f frontend/frontend.log

# Monitor both
tail -f backend/*.log frontend/*.log

# Clear logs
> backend/backend.log
> frontend/frontend.log
```

📚 Next Steps

After successful local deployment:

1. **Explore Features**: Test all functionality

2. **Customize**: Modify colors, branding, content
3. **Develop**: Add new features or modify existing ones
4. **Test**: Run automated tests
5. **Commit**: Push changes to GitHub
6. **Deploy**: Deploy to one.com for public testing

🔗 Additional Resources

- Node.js Documentation (https://nodejs.org/docs/)
- React Documentation (https://react.dev/)
- Vite Documentation (https://vitejs.dev/)
- Express.js Documentation (https://expressjs.com/)
- Material-UI Documentation (https://mui.com/)

Need Help? Check the main deployment guide (../DEPLOYMENT.md) or troubleshooting section.

Last Updated: November 13, 2024 - Phase 2

Note: This localhost refers to localhost of the computer that this guide is running on. To access it remotely or on your local machine, you'll need to deploy the application on your own system following these instructions.