EMSG-Daemon: Copilot/AI Agent Development Guide

# 📘 Project Overview

The EMSG Daemon is a decentralized messaging protocol implementation that simplifies traditional email by removing legacy elements (e.g., Subject, BCC, MIME) and replacing them with a clean structure using identifiers like `user#domain.com`.   
It supports direct and group messaging with persistent threads, decentralized DNS routing, and public key-based authentication.  
Repository: https://github.com/emsg-protocol/emsg-daemon

# 🎯 Project Objectives

- Replace traditional email components with simplified EMSG protocol (TO, CC, Body only)  
- Add support for persistent group messaging using unique `group\_id`  
- Enable decentralized routing via DNS TXT records (e.g., \_emsg.domain.com)  
- Support system messages like group created, joined, left, etc.  
- Public key-based identity and signature verification  
- Modular and production-ready Go codebase with optional REST API

# 🤖 AI Agent Initialization Prompt

You are an expert software development assistant contributing to an open-source Go-based decentralized messaging daemon called EMSG.  
  
This project is not a traditional email server. Instead, it implements a modern and simplified messaging protocol (`user#domain.com` format) with group messaging, public key authentication, and DNS-based message routing.  
  
Ensure the code is modular, production-ready, and follows Go best practices. Avoid legacy SMTP/IMAP logic.

# 🧩 Recommended Components & Files

- main.go — Server entry point  
- router.go — Handle parsing of user#domain.com and DNS TXT record lookup  
- group.go — Group creation, membership management, group\_id persistence  
- message.go — Sending, receiving, and verifying messages  
- auth.go — Public key registration and signature validation  
- storage.go — Message storage (SQLite/Postgres)  
- system.go — System messages (user joined, removed, group created)  
- config.go — Load .env or config file settings  
- api.go (optional) — REST API for admin panel or future web client

# 💡 Copilot & IDE Plugin Prompts

// Prompt: Create a router that parses 'user#domain.com' and fetches DNS TXT records from '\_emsg.domain.com'  
  
// Prompt: Write Go code to validate a digital signature using Ed25519 keys  
  
// Prompt: Implement message routing that uses 'group\_id' to forward messages to all group members  
  
// Prompt: Refactor message storage to use SQLite and support both individual and group messages  
  
// Prompt: Create modular folder structure for daemon services (routing, messaging, groups, auth)

# 📋 Suggested Development Steps

1. Initialize Go project and basic folder layout  
2. Implement `router.go` for decentralized message routing via DNS  
3. Create `auth.go` to handle user key registration and signature verification  
4. Develop `group.go` to manage group creation, membership, and group\_id persistence  
5. Build `message.go` to handle message send/receive, forward, and system messages  
6. Store all data in `SQLite` or `Postgres` via `storage.go`  
7. (Optional) Expose REST endpoints via `api.go` for client integration  
8. Add unit tests for all modules

# 🧪 Test Case Prompts for AI

- Generate tests to verify group membership addition/removal logic  
- Validate routing of messages using DNS TXT record lookup  
- Test cryptographic signing and signature verification  
- Check storage of messages in SQLite and proper retrieval by group\_id or user