# Projektplan – Dezentrales Chat-Programm (BSRN)

## 1. Beschreibung der Implementierung

### 1.1 Grobarchitektur

### Unser Chatprogramm basiert auf drei modularen Prozessen, die über Interprozesskommunikation (IPC) miteinander verbunden sind: dem UI-Prozess, dem Netzwerkprozess und dem Discovery-Dienst.

### Der UI-Prozess stellt eine Kommandozeilenoberfläche bereit. Er nimmt Nutzereingaben entgegen (z. B. Nachrichten, Befehle wie /who oder /away), zeigt eingehende Nachrichten an und verwaltet die Konfigurationsdatei im TOML-Format.

### Der Netzwerkprozess ist für die gesamte Kommunikation nach dem SLCP-Protokoll zuständig. Er sendet und empfängt Nachrichten (JOIN, MSG, IMG, etc.) über UDP und TCP und leitet empfangene Inhalte über IPC an die UI weiter. Bilder werden lokal gespeichert.

### Der Discovery-Dienst reagiert auf WHO-Anfragen im Netzwerk und sendet eine Liste bekannter Clients zurück. Er wird beim Start des Programms automatisch einmal pro Gerät gestartet und stellt sicher, dass keine doppelte Instanz läuft.

### Alle Komponenten greifen auf eine zentrale Konfigurationsdatei zu, die z. B. Benutzernamen, Ports und Autoreply-Text enthält. Die Architektur ist bewusst dezentral gehalten, einfach erweiterbar und erlaubt klare Testbarkeit der einzelnen Module.

### Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält. KI-generierte Inhalte können fehlerhaft sein.

### 1.2 Verwendete Technologien

- Programmiersprache: Python  
- Entwicklungsumgebung: Visual Studio Code  
- Versionskontrolle: Git über GitHub  
- Kommunikation: TCP/UDP über Python socket  
- Konfiguration: TOML-Dateien  
- Dokumentation: Doxygen für Python  
- IPC: multiprocessing (Queue, Pipe)

## 2. Projekt- und Zeitplan

### 2.1 Arbeitspakete und Verantwortlichkeiten

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Verantwortung | Aufgaben (Kurzfassung) | Technologien |
| Nabil | Netzwerk (SLCP) | Nachrichten senden/empfangen, SLCP-Befehle implementieren | socket, TCP, UDP |
| Aset | Discovery | Broadcast-Dienst WHO/KNOWNUSERS, Teilnehmerliste pflegen | UDP-Broadcast, socket |
| Effe | Benutzeroberfläche & Konfiguration | CLI-Interface, Konfig-Dateien, Autoreply | argparse, toml |
| Daniyal | Prozesssteuerung & IPC | Prozesse starten/verbinden, IPC, Discovery nur einmal | multiprocessing, IPC |

### Aufgaben von Nabil

* Implementierung der SLCP-Kommandos: JOIN, LEAVE, MSG, IMG
* Senden/Empfangen über UDP (JOIN, LEAVE, WHO)
* Senden/Empfangen über TCP (MSG, IMG)
* Einhalten des Nachrichtenformats inkl. Encoding (UTF-8) und max. Länge
* Verwaltung der bekannten Peers und Zuordnung von IP/Ports

### Aufgaben von Aset

* Aufbau des Discovery-Diensts als separater Prozess
* Empfang von WHO-Anfragen über UDP-Broadcast
* Antwort mit KNOWUSERS über Unicast
* Pflege einer lokalen Liste bekannter Teilnehmer mit Zeitstempel
* Sicherstellen, dass Discovery nur einmal pro Gerät läuft (z. B. via Socketbindung)

### Aufgaben von Effe

* Design und Umsetzung des CLI-Menüs zur Bedienung
* Einlesen und Speichern der Konfigurationsdatei (TOML)
* Anzeige von Nachrichten im Terminal mit zeitlichem Verlauf
* Autoreply-Logik bei gesetztem Abwesenheitsstatus
* GUI-Prototyp

### Aufgaben von Daniyal

* Startskript zum Erzeugen der drei Hauptprozesse (UI, Netzwerk, Discovery)
* Implementierung der Kommunikation über multiprocessing.Queue oder Pipe
* Signalhandling & sauberes Beenden aller Prozesse
* Verbindungsmanagement zwischen UI & Netzwerkprozess
* Verhinderung mehrfach gestarteter Discovery-Dienste

### 2.2 Meilensteine

- M1: Architektur und Grobplanung (10.05.2025)  
- M2: Grundfunktionen implementiert: Netzwerk, Discovery, UI (20.05.2025)  
- M3: Interprozesskommunikation aktiv, erste Ende-zu-Ende-Kommunikation (27.05.2025)  
- M4: Bildübertragung, Konfiguration, Autoreply (07.06.2025)  
- M5: Doxygen-Dokumentation fertiggestellt (14.06.2025)- M6: Generalprobe Präsentation + Abgabeprojekt (20.06.2025)

Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.