INTERNATIONAL UNIVERSITY
School of Computer Science and Engineering
(Sem 1, 2025 - 2026)

STUDENT PRESENTATION

MANGAHUB - BY GROUP 05

Net - Centric Programming

Lê Hưng - ITCSIU22271

Đàm Nguyễn Trọng Lễ- ITITIU22240

CONTENT

INTRODUCTION

In this part, we briefly introduce to the content, explain about the meaning of “SOCKET PROGRAMMING” and other terminologies.

Team 5: Task Distribution

Lê Hưng:

- Database design
- Manga CRUD
- UDP - Notification service
- Sync manga with external APIs
- GRPC for manga query

Trọng Lễ:

- Authentication
- TCP - Sync Progress
- GRPC for sync progress
- Websocket chat system
- Automation Test
- DevOps

PROBLEM & MOTIVATION

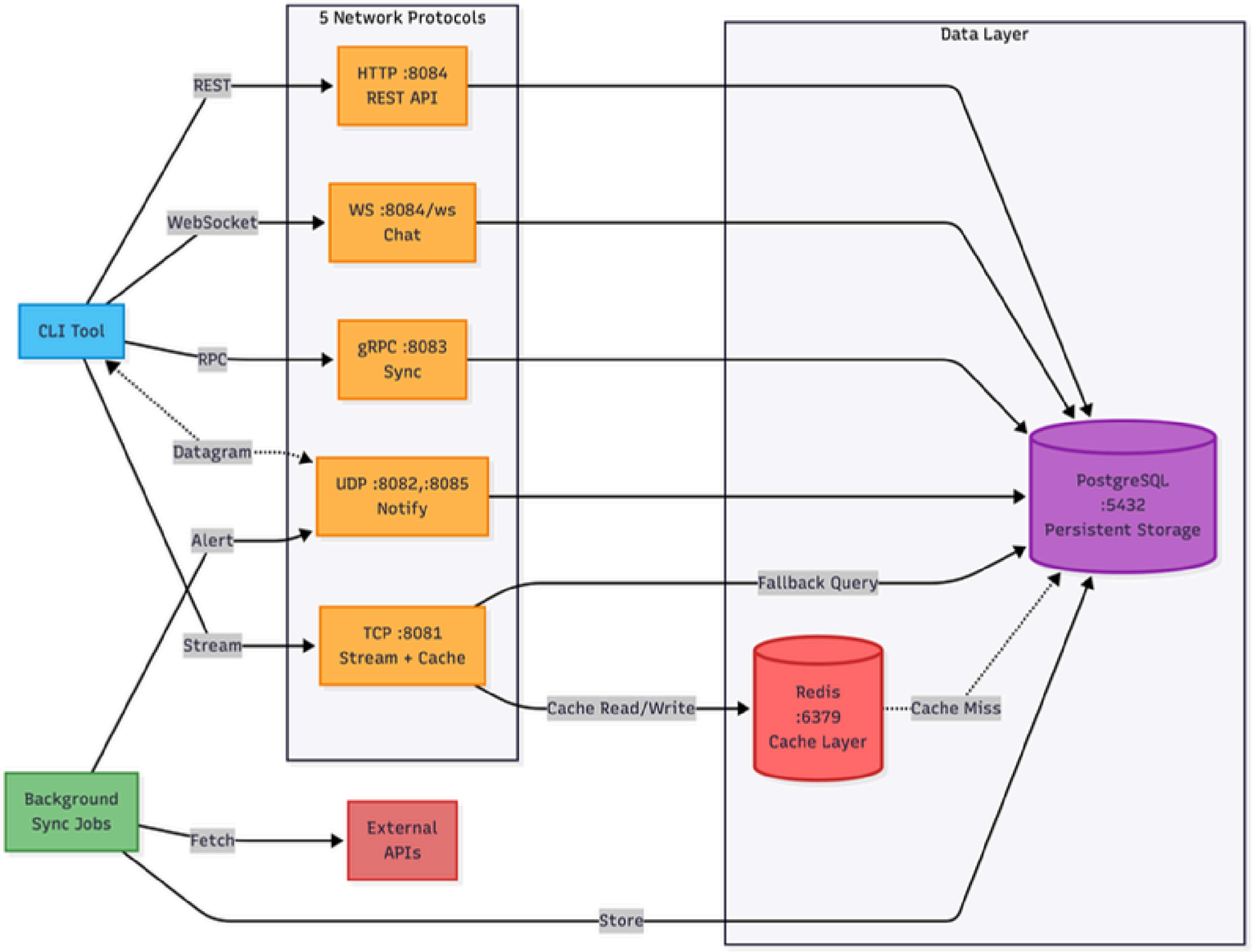
- Pain points: tracking manga progress across many series and sources:
- Manga library & progress tracking.
- Real-time sync, notifications, chat.
- Automatic data sync from MangaDex & AniList
- Goal: unified CLI-based manga hub with real-time features.

PROJECT OBJECTIVES

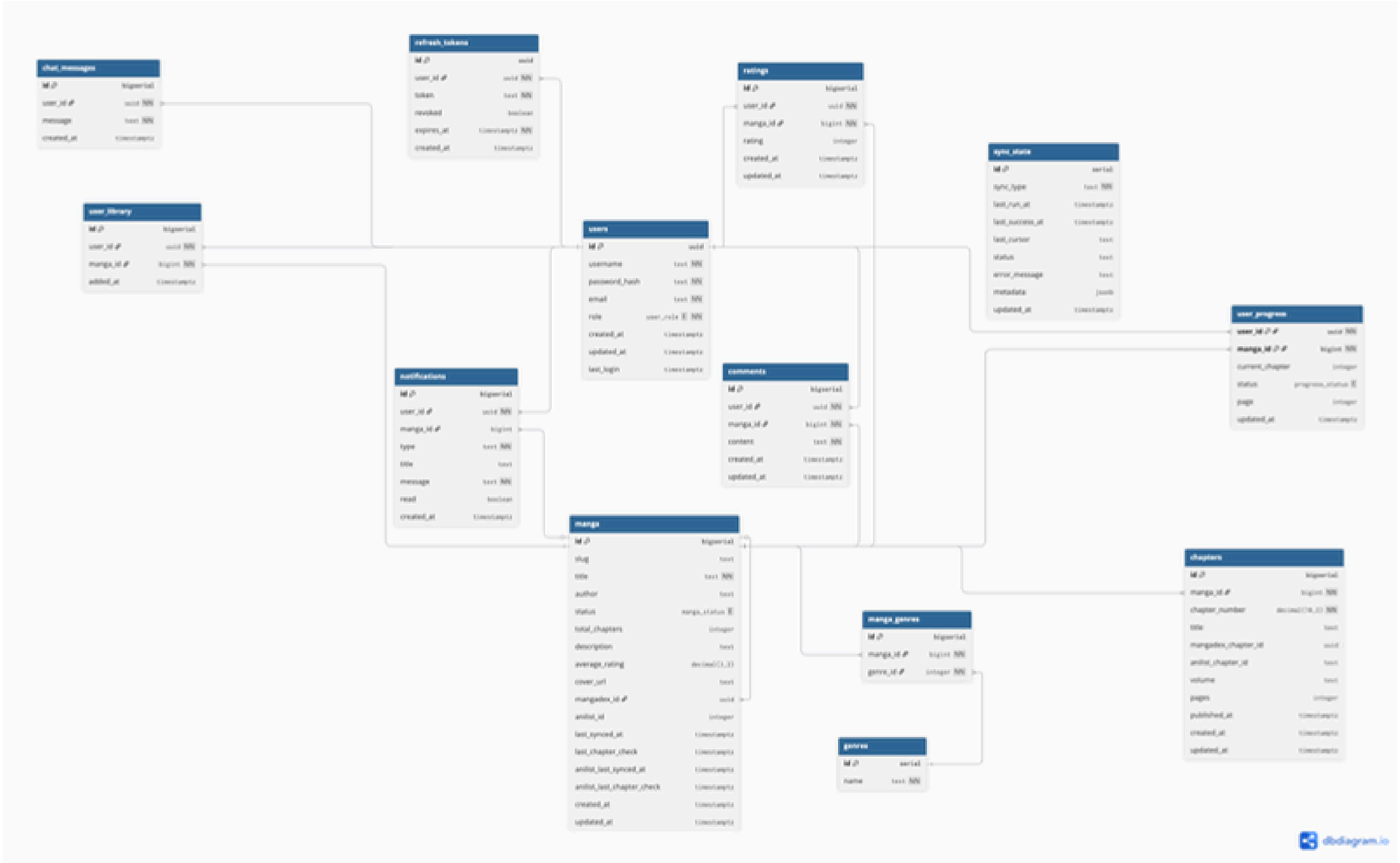
- Learn and apply 5 network protocols in Golang.
- Build end-to-end system: CLI → services → database/cache.
- Real-world skills: concurrency, distributed systems, API integration.

SYSTEM ARCHITECTURE

SYSTEM OVERVIEW



DATABASE SCHEMA



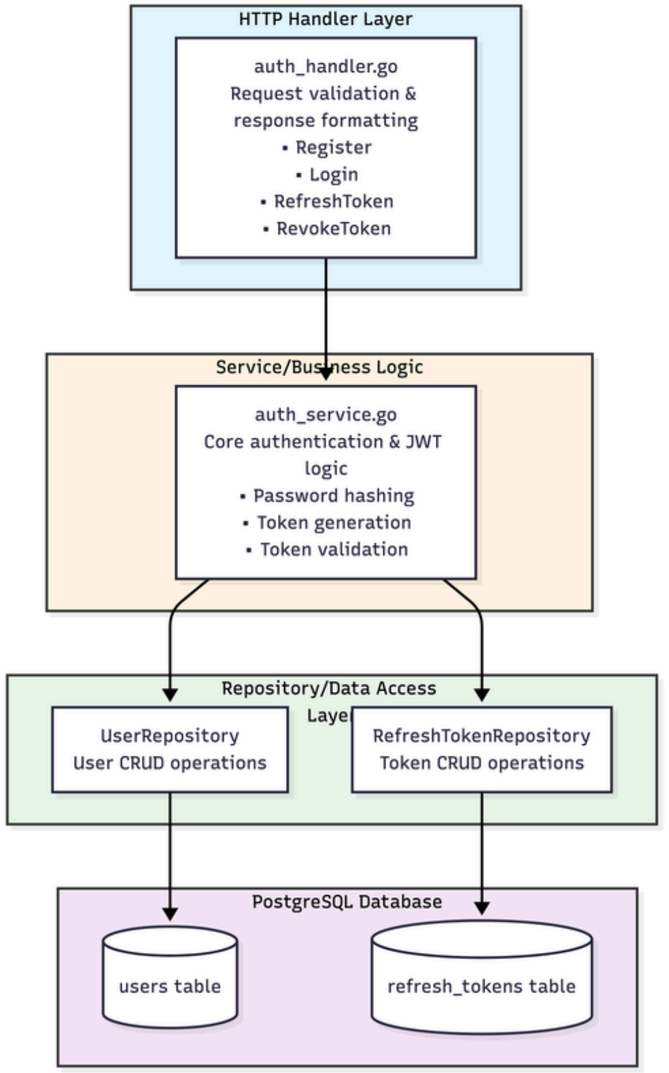
SERVICE WORKFLOWS

HTTP-REST API FOR MANGA CRUD

Clear layered HTTP architecture: request flows from the HTTP layer through handler, service, repository, down to PostgreSQL, which makes responsibilities easy to reason about.

→ This separation of concerns improves testability: handlers, services, and repositories can be unit-tested independently with mocks instead of hitting the real database.

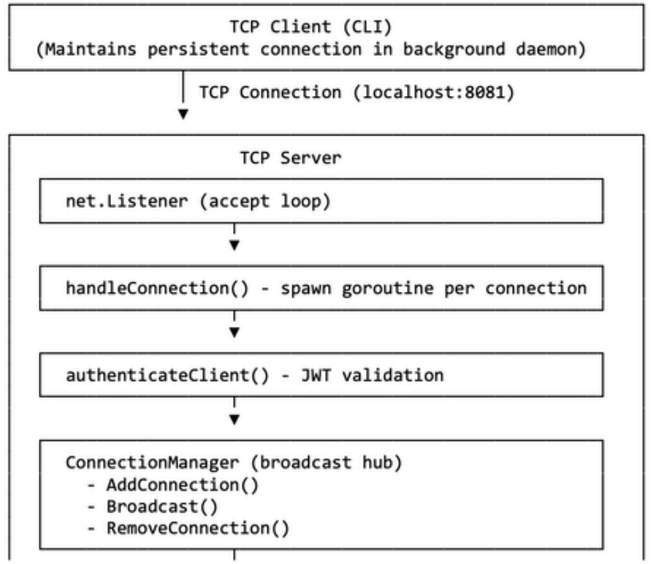
AUTHENTICATION LOGIC

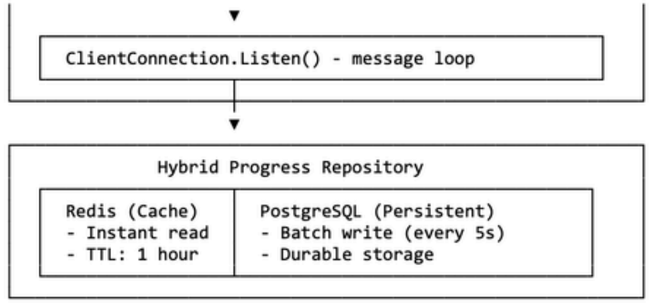


- Input validation (username 3-50 chars, password ≥8 chars, valid email)
- Duplicate username/email check
- Refresh token stored in database with expiry
- Parse and validate JWT signature using HMAC-SHA256
- Inject user context into request (userID, role, scopes)
- **Token Refresh** (/auth/refresh) ⟶ Prevents token reuse attacks

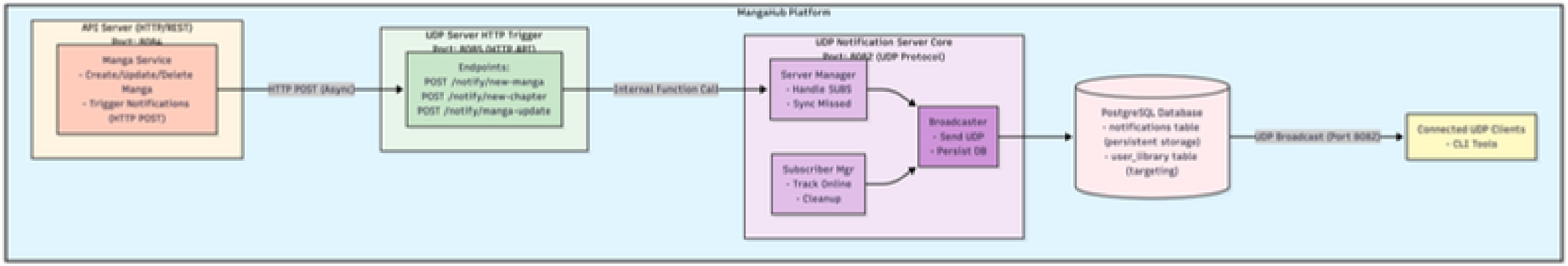
TCP - SYNC READING PROGRESS

- Protocol: Raw TCP with custom JSON-based wire protocol
- Storage: Hybrid (Redis cache + PostgreSQL persistence)
- Connection Model: Long-lived, persistent connections with heartbeat
- Authentication: JWT validation at connection establishment
- Concurrency: Goroutine-per-connection model with broadcast channels





UDP - REAL TIME NOTIFICATION



Why Two Ports?

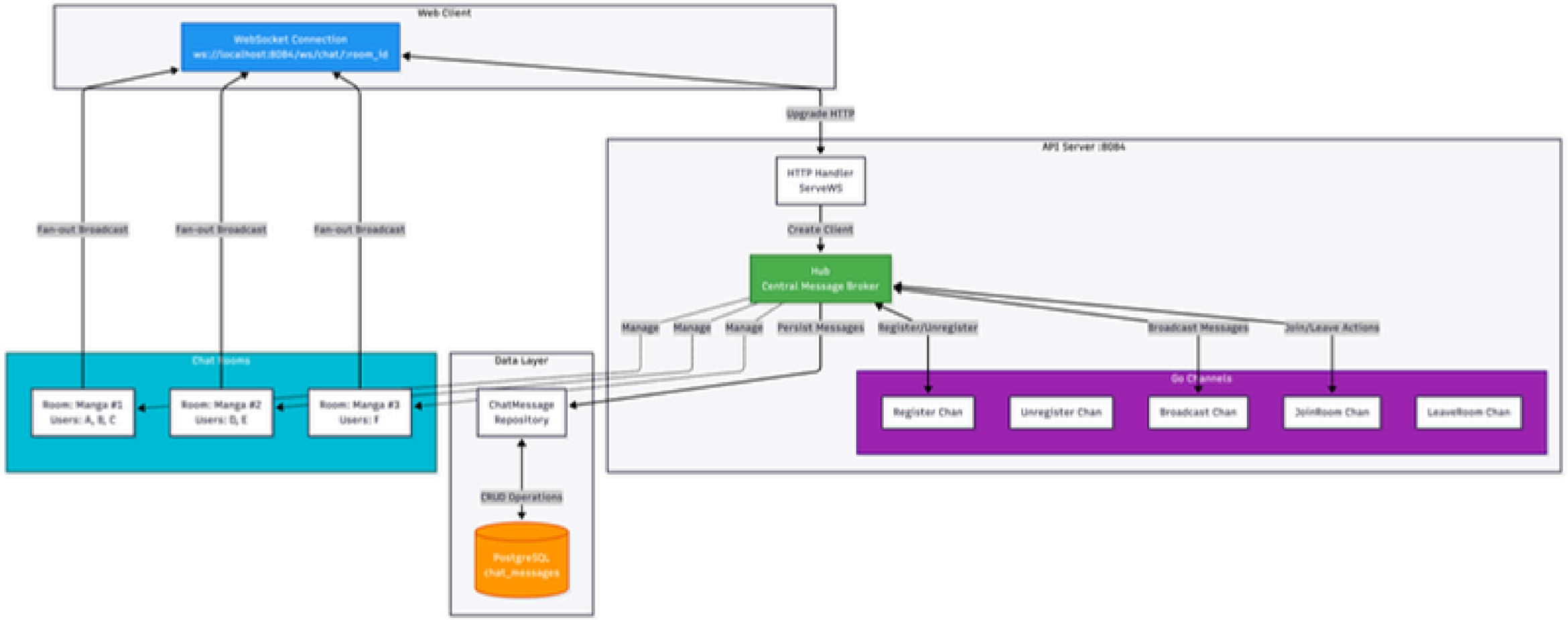
Port	Protocol	Purpose	Accessed By
8082	UDP	Client connections & real-time notifications	CLI clients, end users
8085	HTTP	Internal trigger API for services	API server, MangaDex sync, AniList sync

Port 8085 (HTTP Trigger Interface)

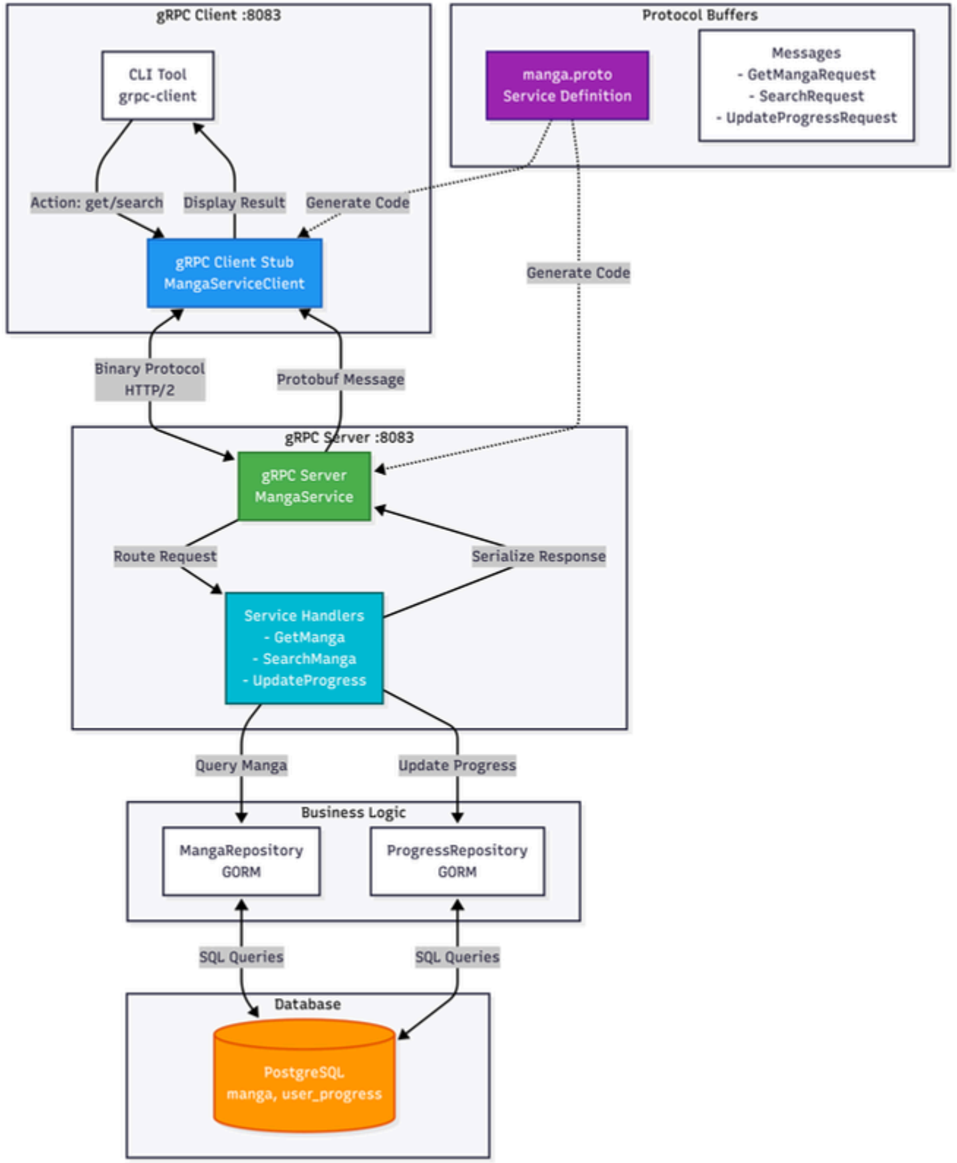
Role: HTTP-to-UDP Bridge - Internal API Gateway

This port exposes HTTP endpoints that internal services use to trigger UDP broadcasts:

WEBSOCKET - CHAT SYSTEM

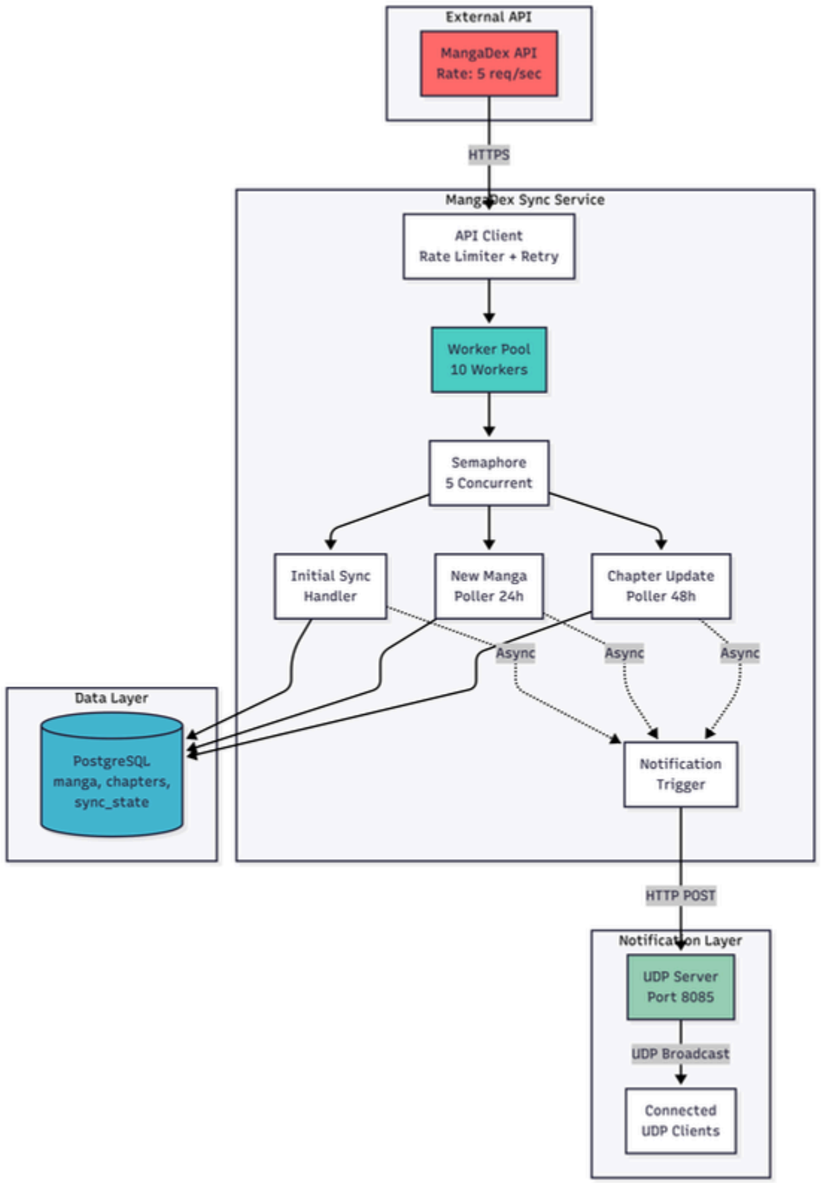


GRPC - INTERNAL CALL




```
CLI Client
$ mangahub grpc search "Query….." --limit 10
        |
        ▼
   gRPC Call (HTTP/2 over TCP)
        |
        ▼
gRPC Server (Port 8083)
  MangaServiceServer (implements proto interface)
  - GetManga(id) → pb.Manga
  - SearchManga(query) → pb.MangaSearchResponse
  - UpdateProgress(data) → pb.ProgressUpdateResp
        |
        ▼
  MangaService (business logic layer)
  - Validates input
  - Orchestrates data fetching
  - Converts models to protobuf
        |
        ▼
Data Layer (Multi-Source)
  PostgreSQL DB          | External APIs
  - Local manga          | - MangaDex API
  - User progress        | - AniList API
                         | - MyAnimeList
```

MANGA - SYNC DATA PERSISTENTLY



- Problem: Processing 200 manga sequentially takes 40 seconds (0.2s per manga).

--> Solution: Fixed-size worker pool that reuses goroutines.

- Problem: MangaDex API allows max 5 requests/second. Worker pool with 10 goroutines could exceed this.

-->Solution: Semaphore pattern to limit concurrent API calls.

- Problem: Even with semaphore, bursty traiic could temporarily exceed rate limits.

→ Solution: Token bucket algorithm from golang.org/x/time/rate.

INTERNATIONAL UNIVERSITY
School of Computer Science and Engineering
(Sem 1, 2025 - 2026)

THANK YOU

FOR YOUR ATTENTION - FEEL FREE TO ASK FOR MORE DETAILS

REFERENCES:

- https://pkg.go.dev/net
- https://github.com/headtomatoes/mangahub