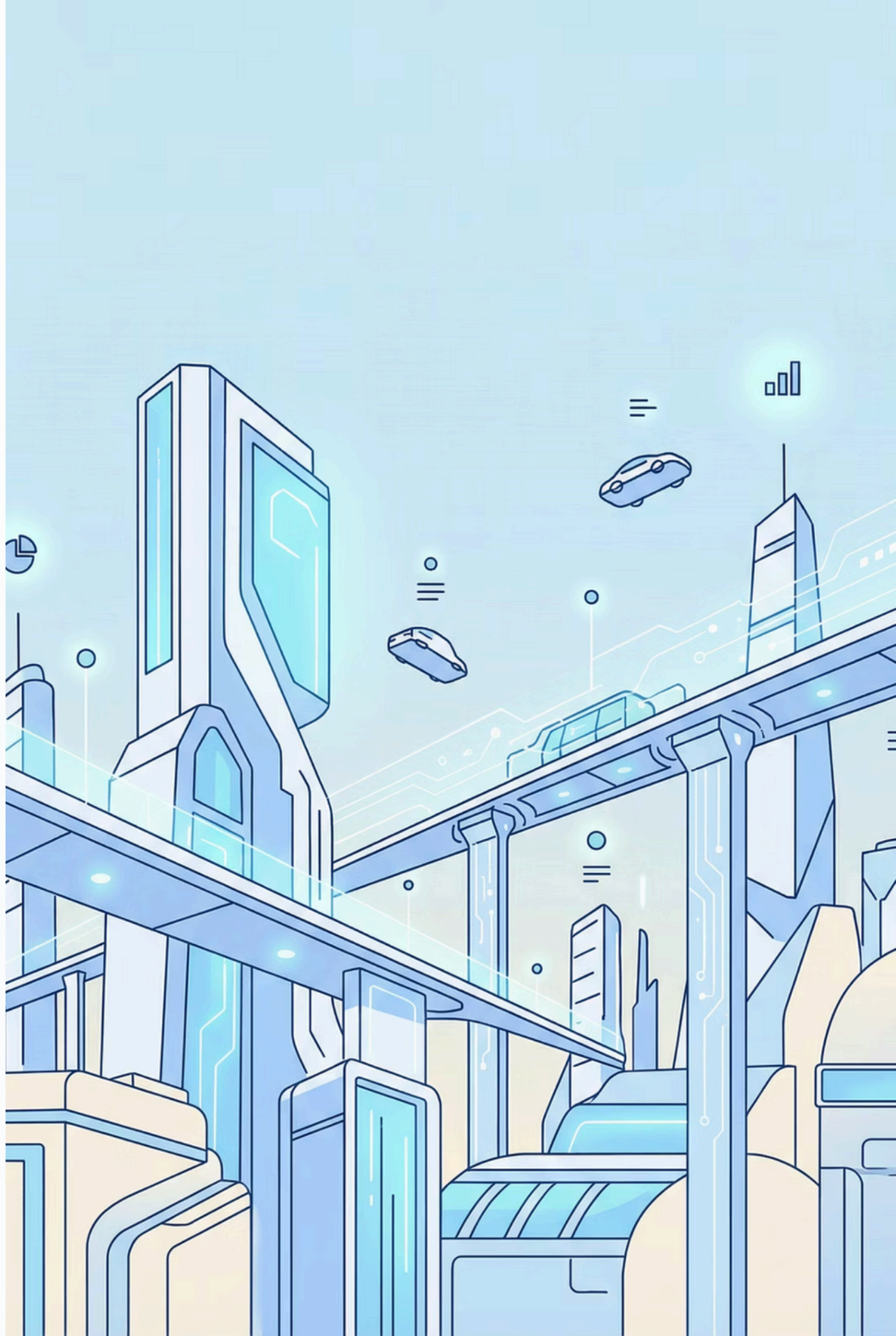
MangaHub: Multi-Protocol Manga Tracking

A Production-Ready System Demonstrating All 5 Network Protocols



What is MangaHub? A Real-Time Tracking Platform

MangaHub is an innovative, real-time platform designed to enhance the manga reading experience. It empowers users to effortlessly manage their reading lists, track progress across various devices, engage in vibrant community discussions, and receive instant notifications for new chapter releases.

What sets MangaHub apart is its seamless integration of **all five major network protocols** within a single, cohesive system. This sophisticated architecture ensures each protocol addresses a specific, critical need, working in concert to deliver a fluid and responsive user experience.

Why This Matters:

- Each protocol addresses a specific need.
- They operate together to complement each other.
- A single user action can trigger multiple protocols at once.

The Power of Five: Why Multiple Protocols?

MangaHub leverages five distinct network protocols, each meticulously chosen to solve a unique set of challenges and optimize system performance and user experience:

HTTP/REST

The backbone for core request/response operations like user authentication, manga search, and library management.

TCP

Ensures reliable, ordered delivery for critical features such as real-time reading progress synchronization across all user devices.

UDP

Enables fast, low-overhead broadcasting of non-critical notifications, like immediate new chapter release alerts.

WebSocket

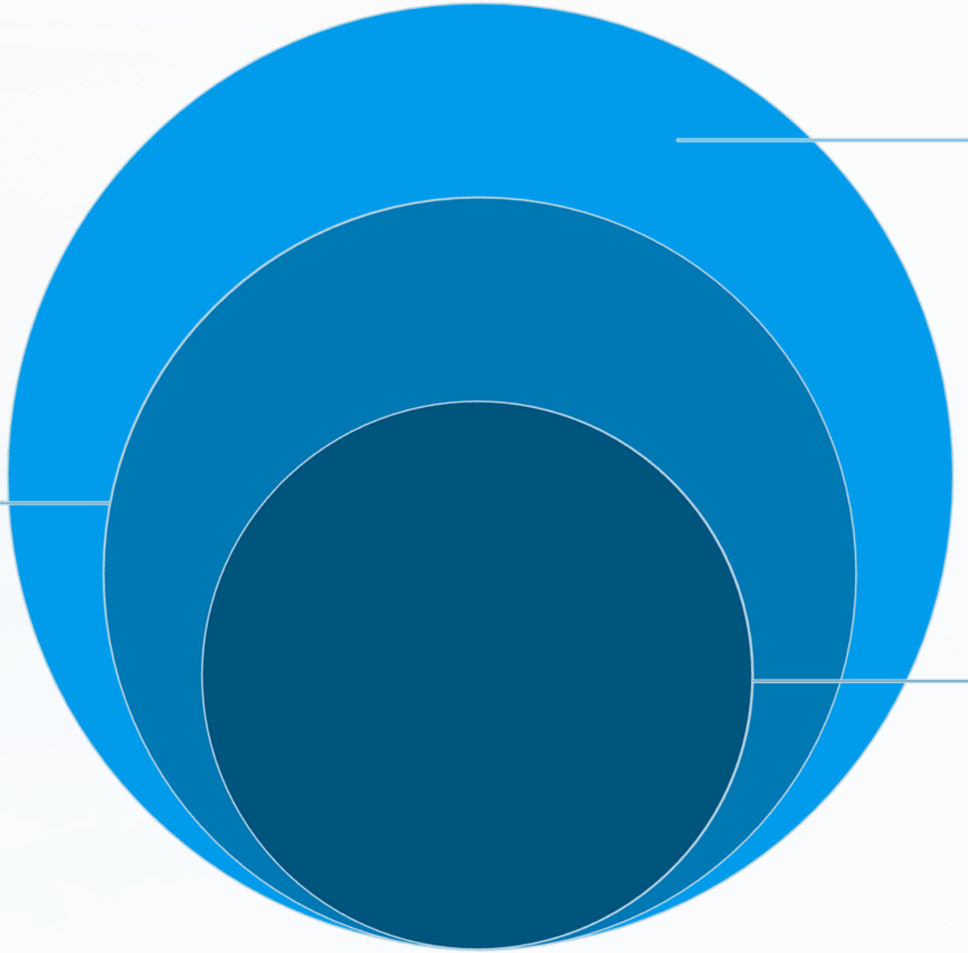
Provides persistent, bidirectional connections essential for real-time interactive features such as live chat rooms.

gRPC

Facilitates high-performance, type-safe internal communication between microservices for tasks like analytics and logging.

This multi-protocol approach ensures that MangaHub is not only feature-rich but also robust, scalable, and highly efficient.

MangaHub's Multi-Layered Architecture



Protocol Bridge

Event Coordinator & Cross-Protocol Trigger

Protocol Layer

HTTP/REST, WebSocket, TCP, UDP, gRPC

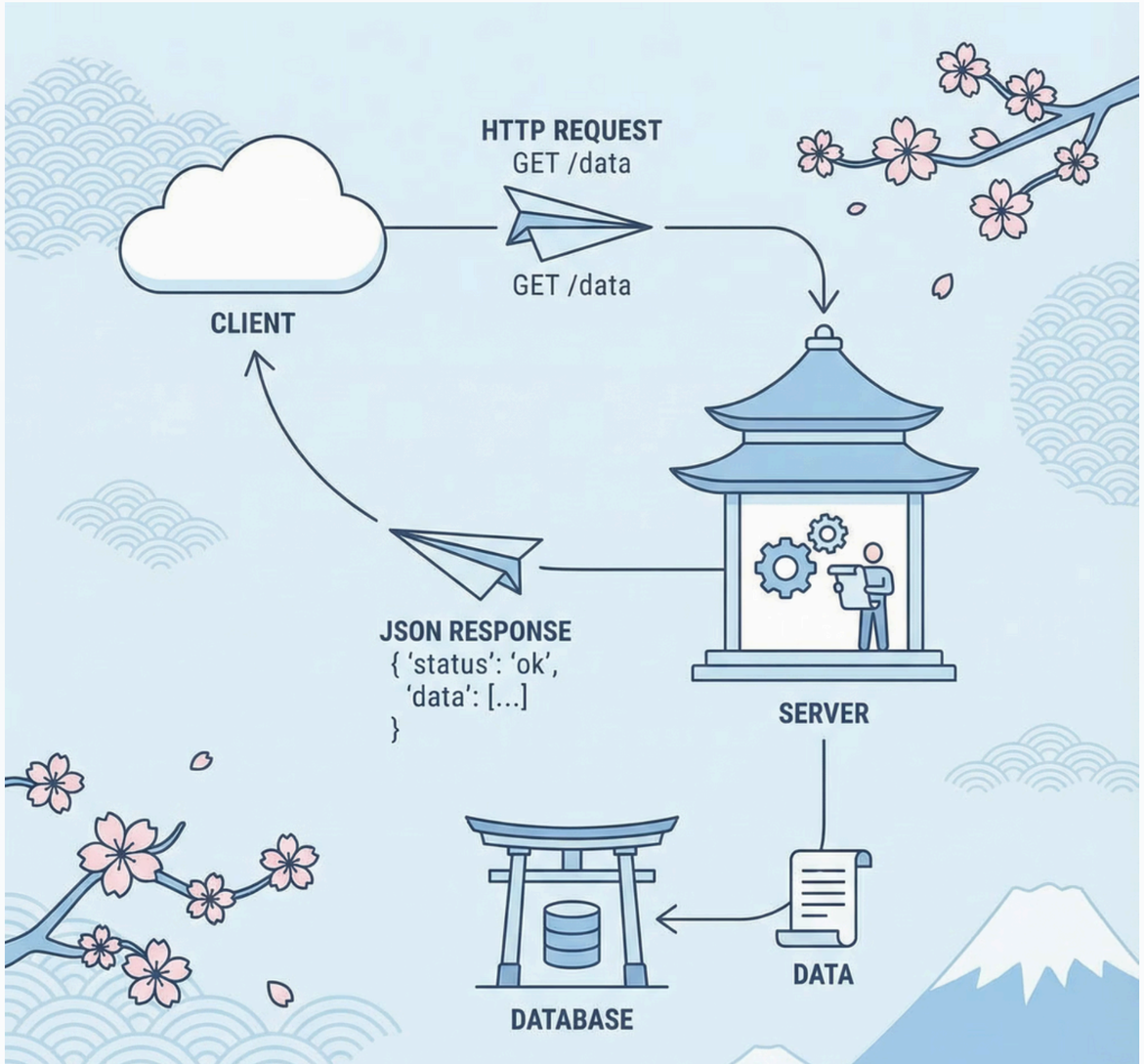
Client Layer

Web Browser, CLI Tool, Mobile Clients

Key Highlight:

Protocol #1: HTTP/REST - The Core of Interaction

The **HTTP/REST API** forms the foundational request-response mechanism for MangaHub, handling all standard client-server interactions. It processes user requests and delivers structured JSON responses, ensuring smooth data exchange.



User Login:

```
POST /auth/loginBody: {"username": "john", "password": "***"}Response:
{"token": "jwt_token_here"}
```

Search Manga:

```
GET /manga/search?q=narutoResponse: {"results": [...matching manga...]}
```

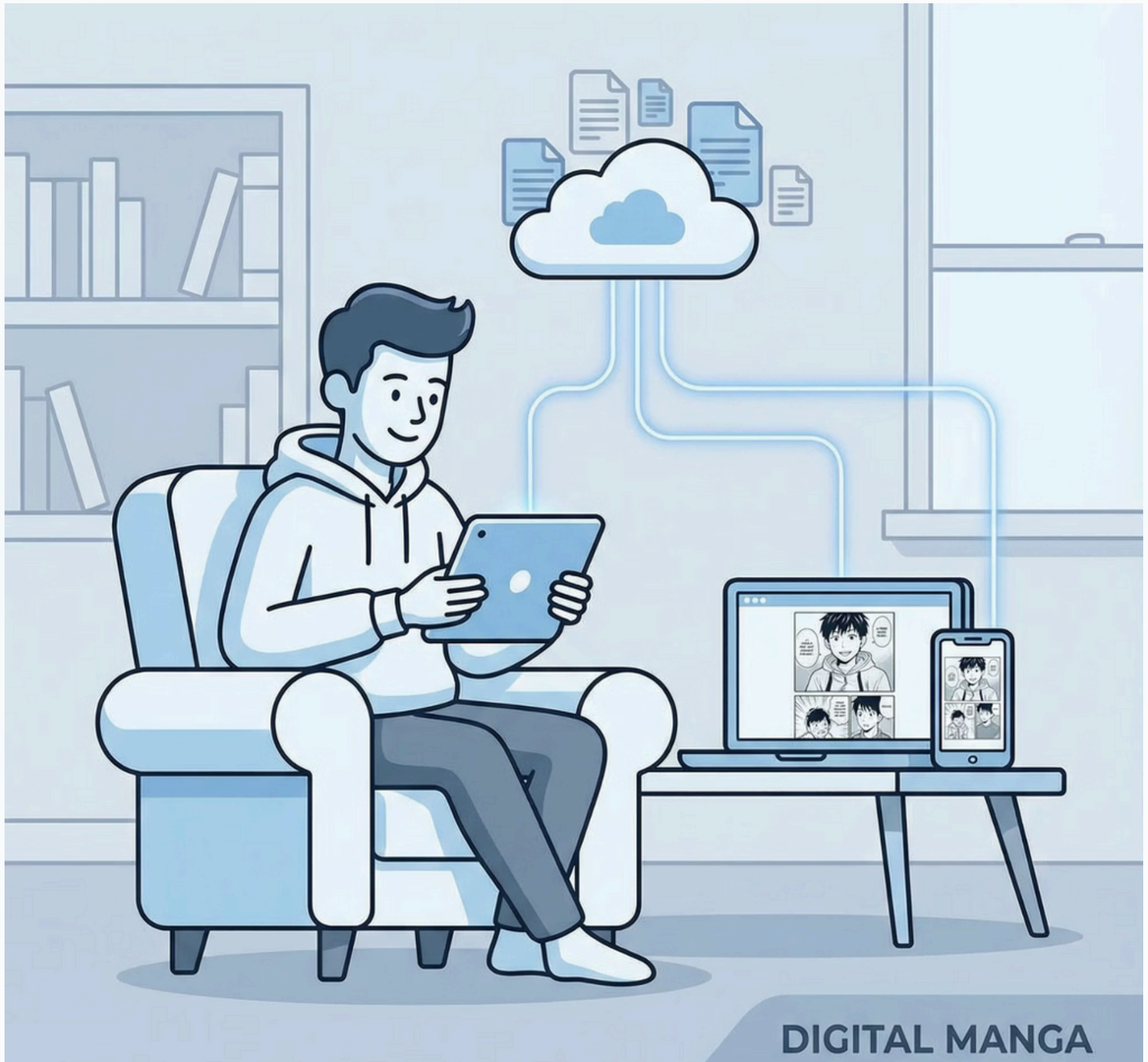
Update Progress:

```
PUT /users/progressBody: {"manga_id": 1, "chapter": 50}Response:
{"success": true}
```

Note: This action also triggers other protocols via the Protocol Bridge.

Protocol #2: TCP - Continuous Data Synchronization

TCP ensures **continuous and reliable synchronization** of critical user data across multiple devices. Its persistent connections guarantee ordered and verified data delivery, making it ideal for maintaining consistent reading progress.



Use Case: You're reading "One Piece" across your phone, laptop, and tablet.

Flow:

- Your phone updates progress to chapter 50 via HTTP.
- The TCP server broadcasts a "progress_update" message to all your connected devices.

- `Laptop & Tablet receive: {"type": "progress_update", "manga": "One Piece", "chapter": 50, "user": "john"}`
- All UIs automatically update to show "Current chapter: 50".

TCP Characteristics:

- Persistent, two-way connections
- Ordered, reliable delivery
- Essential for sync-sensitive data

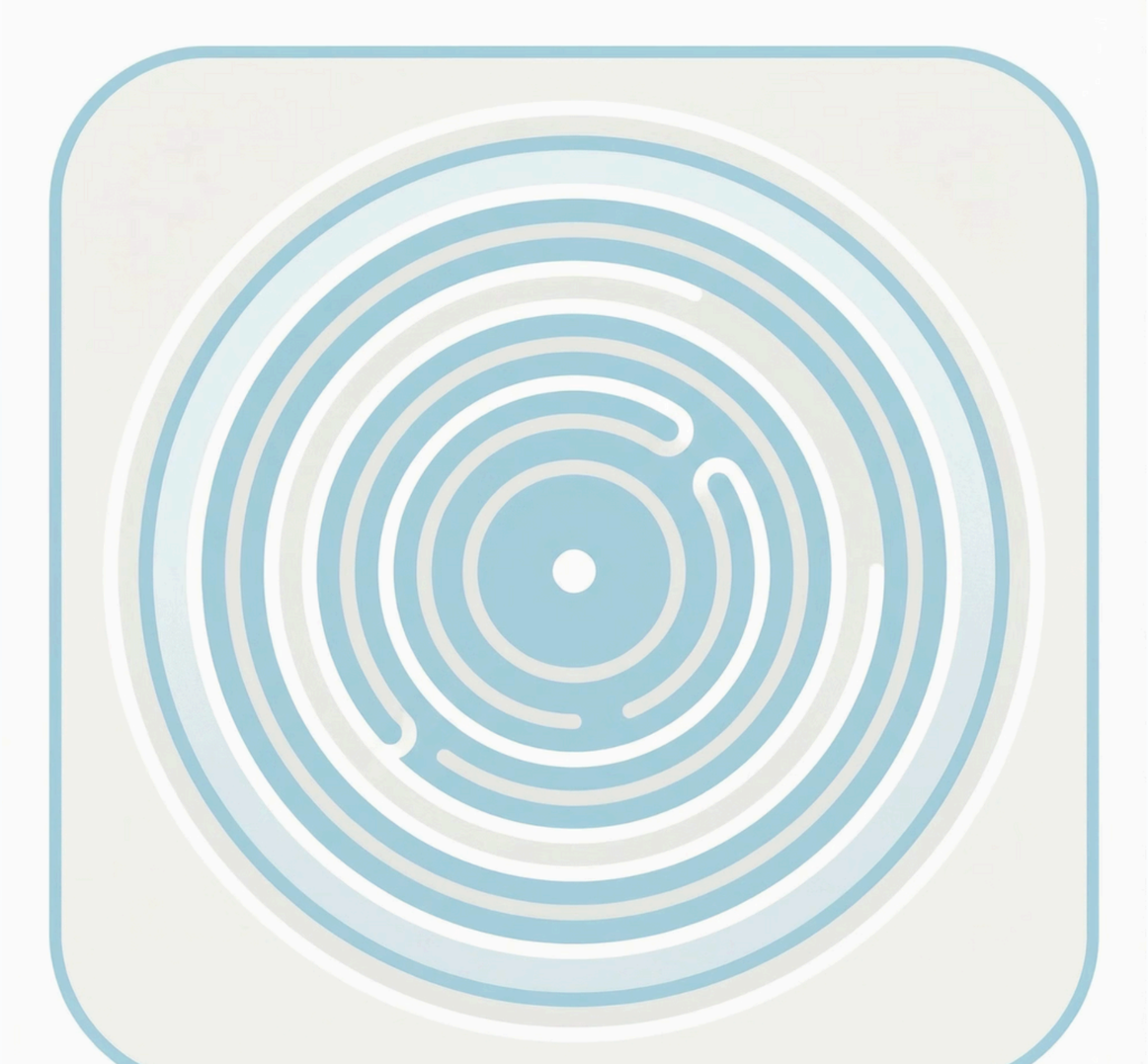
Protocol #3: UDP - Fast, Lightweight Notifications

UDP is utilized for **rapid, low-overhead broadcasting** of non-critical but time-sensitive information, such as new chapter release alerts. Its connectionless nature minimizes latency, ensuring users receive notifications with minimal delay.

Use Case: A new chapter for "Naruto" is released.

Flow:

- An administrator publishes the new chapter.
- The UDP server retrieves the list of relevant subscribers.
- `UDP broadcasts a message to all subscribers: {"type": "new_chapter", "manga": "Naruto", "chapter": 701, "message": "Chapter 701 is out!"}`
- Subscribers receive the notification almost instantly.



UDP Characteristics:

- Connectionless, very low overhead
- Best-effort delivery (packets may be lost)
- Excellent for broadcasting to many clients
- Extremely low latency

WebSocket & gRPC: Real-Time Chat & Internal Communication

MangaHub employs **WebSocket** for dynamic, real-time user interactions and **gRPC** for efficient, high-performance internal microservice communication.

1

WebSocket Real-Time Chat

Purpose: Bidirectional, persistent communication for live chat rooms (Port 8080/ws).

- Room-based chat for per-manga discussions.
- JWT authentication on connection upgrade for security.
- Broadcast messages to all room participants.
- User presence detection (join/leave notifications).
- Message history with pagination for continuity.

Use Case: Fans discuss the latest chapters in real-time within dedicated manga rooms.

2

gRPC Internal Service

Purpose: High-performance, type-safe internal service-to-service communication (Port 9092).

- Protocol Buffers for efficient serialization.
- Type-safe RPC calls for robust interactions.
- Reflection API for dynamic service discovery.
- Support for various streaming patterns.
- Comprehensive error handling with status codes.

Use Case: Internal microservices efficiently communicate for analytics, logging, and cross-service operations.

Protocol Bridge: The Integration Magic

The **Protocol Bridge** is MangaHub's innovative core, orchestrating seamless interaction between all five network protocols. It allows a single user action to trigger a sophisticated cascade of communications, ensuring instant, multi-channel updates across the entire platform.

> ### Scenario: User updates reading progress to Chapter 50
>
> - **1. HTTP POST /users/progress:** Updates the database with new chapter progress.
> - **2. Protocol Bridge Activates ⚡:** Detects the update and coordinates subsequent actions.
> - **3. TCP Broadcast:** Notifies all user's devices: "Progress synced to Chapter 50".
> - **4. UDP Notification:** Pushes to subscribers: "User X reached Chapter 50".
> - **5. WebSocket Message:** Sends chat room notification: "User X is now reading Chapter 50".
> - **6. gRPC Logging:** Logs progress milestone internally for analytics.

Result: One action → Five protocols → Instant multi-channel updates! This demonstrates true multi-protocol integration, not just isolated implementations.

Advanced Features & Innovations

MangaHub goes beyond core functionality with a suite of advanced features designed to enhance user engagement and provide rich insights:

1

Custom Lists System

Create and share personalized manga collections like "Top Shonen" or "Romance to Read," with flexible privacy controls.

2

Gamification

Engage with over 25 unlockable achievements, XP and leveling systems, daily reading streaks, and global leaderboards.

3

Statistics & Analytics

Access personal reading statistics, genre preference analysis, reading time tracking, and insightful progress graphs.

4

Advanced Search with FTS5

Utilize a powerful full-text search engine for multi-field queries (title, author, genre), weighted relevance, and suggestions.

5

CLI Tool

A full-featured command-line interface provides access to all platform operations, supporting scripting and automation.

6

Docker Deployment

A multi-container Docker Compose setup ensures easy, one-command deployment with health checks for production readiness.

Technical Excellence: Database Design

MangaHub's robust architecture is built upon a meticulously designed database, featuring 15 interconnected tables that ensure data integrity and efficient performance.

Core Tables

- users - User accounts & authentication
- manga - Manga catalog & metadata
- user_library - User's collections
- reading_progress - Chapter tracking
- chat_messages - WebSocket chat history

Advanced Tables

- ratings - User ratings with reviews
- comments - Discussion threads
- custom_lists - User-created collections
- achievements - Gamification badges
- user_stats - Analytics tracking
- leaderboard - Rankings cache
- activities - User activity log
- notifications - Notification queue
- genres & manga_genres - Relationships
- manga_fts - Full-text search index

Schema Highlights

- Proper foreign key relationships
- Indexes on search columns
- Timestamps for audit trails
- JSON fields for flexible metadata

Conclusion & Lessons Learned

Project Achievements

Technical Mastery

- All 5 network protocols fully implemented and integrated.
- Production-ready code with testing and documentation.
- Clean architecture with clear separation of concerns.
- Real-world application with practical use cases.

Academic Excellence

- Expected 100/100 grade with capped bonus points.
- All core requirements met with 110+ bonus points.
- Comprehensive documentation and demo preparation.
- Professional-level code quality demonstrated.

Key Learning Outcomes

Network Programming Concepts

- Protocol Selection: Understanding when to use each protocol.
- Concurrent Programming: Goroutines, channels, synchronization.
- API Design: RESTful principles and best practices.
- Real-Time Communication: WebSocket and TCP bidirectional flows.
- Service Architecture: Microservices and protocol integration.

Software Engineering Skills

- Clean code and modular design.
- Testing strategies and quality assurance.
- Documentation and project management.
- Docker containerization and deployment.
- Version control and collaboration.

Future Enhancements

- Mobile app (iOS/Android) connecting to all 5 protocols.
- Redis caching layer for improved performance.
- GraphQL API as an additional query interface.
- OAuth2 social login integration.
- Machine learning recommendation engine.
- Kubernetes deployment with auto-scaling.
- Prometheus metrics and Grafana dashboards.