MangaHub - Use Case Specification (Revised Scope)

1. Introduction

1.1 Purpose

This document defines the functional requirements for MangaHub through detailed use case specifications. The scope has been adjusted for 12-week implementation by junior/senior level students with limited Go experience.

1.2 Scope

MangaHub is a simplified manga tracking system demonstrating network programming concepts through practical implementation of TCP, UDP, HTTP, gRPC, and WebSocket protocols.

1.3 System Overview

The system provides basic manga tracking, reading progress synchronization, and community features while maintaining realistic complexity for academic implementation.

2. Actor Definitions

2.1 Primary Actors

- **Manga Reader**: End users who track manga reading progress
- **Chat User**: Users participating in real-time discussions
- **System Administrator**: Staff managing basic system operations

2.2 Secondary Actors

- **TCP Client**: Applications connecting to sync server
- **UDP Client**: Applications receiving notifications
- **WebSocket Client**: Browser connections for real-time chat
- **External APIs**: MangaDx API for additional manga data

3. Core Use Cases

3.1 User Management

UC-001: User Registration

Primary Actor: Manga Reader
Goal: Create a new user account
Preconditions: None
Postconditions: User account is created

Main Success Scenario: 1. User provides username, email, and password

2. System validates input format and uniqueness
3. System hashes password using bcrypt
4. System creates user record in SQLite database
5. System returns success confirmation

Alternative Flows: - A1: Username already exists - System returns error message

- A2: Invalid email format - System requests valid email
- A3: Weak password - System displays password requirements

UC-002: User Authentication

Primary Actor: Manga Reader
Goal: Login to access personalized features
Preconditions: User has valid account
Postconditions: User is authenticated with JWT token

Main Success Scenario: 1. User provides username/email and password

2. System validates credentials against database
3. System generates JWT token with user information
4. System returns token for subsequent requests
5. User can access protected endpoints

Alternative Flows: - A1: Invalid credentials - System returns authentication error

- A2: Account not found - System suggests registration

3.2 Manga Discovery and Management

UC-003: Search Manga

Primary Actor: Manga Reader
Goal: Find manga series using search criteria
Preconditions: System has manga database populated
Postconditions: Relevant manga results are displayed

Main Success Scenario: 1. User enters search query (title or author)

2. System queries SQLite database using LIKE patterns
3. System applies basic filters (genre, status) if provided
4. System returns paginated results with basic information
5. User can select manga for detailed view

Alternative Flows: - A1: No results found - System displays "no results" message

- A2: Database error - System logs error and returns generic message

UC-004: View Manga Details

Primary Actor: Manga Reader
Goal: Access detailed information about specific manga
Preconditions: Manga exists in database
Postconditions: Complete manga information is displayed

Main Success Scenario: 1. User selects manga from search results or direct URL

2. System retrieves manga details from database
3. System displays title, author, genres, description, chapter count
4. System shows user's current progress if logged in
5. User can add manga to library or update progress

UC-005: Add Manga to Library

Primary Actor: Manga Reader
Goal: Add manga to personal reading library
Preconditions: User is authenticated, manga exists
Postconditions: Manga is added to user's library

Main Success Scenario: 1. User clicks "Add to Library" from manga details

2. System presents status options (Reading, Completed, Plan to Read)
3. User selects initial status and current chapter

4. System creates user_progress record in database

5. System confirms addition and updates UI

Alternative Flows: - A1: Manga already in library - System offers to update status

- A2: Database error - System logs error and shows retry option

UC-006: Update Reading Progress

Primary Actor: Manga Reader
Goal: Track current reading progress
Preconditions: Manga is in user's library
Postconditions: Progress is updated locally and broadcasted

Main Success Scenario: 1. User updates current chapter number

2. System validates chapter number against manga metadata

3. System updates user_progress record with timestamp

4. System triggers TCP broadcast to connected clients

5. System confirms update to user

Alternative Flows: - A1: Invalid chapter number - System shows validation error

- A2: TCP server unavailable - System updates locally, queues broadcast

3.3 Real-time Progress Synchronization

UC-007: Connect to TCP Sync Server

Primary Actor: TCP Client
Goal: Establish connection for real-time progress updates
Preconditions: TCP server is running
Postconditions: Client is connected and registered

Main Success Scenario: 1. Client initiates TCP connection to server

2. Server accepts connection and creates goroutine handler

3. Client sends authentication message with user credentials

4. Server validates user and adds connection to active list

5. Server confirms successful registration

Alternative Flows: - A1: Authentication fails - Server closes connection

- A2: Server at capacity - Server rejects connection with error

UC-008: Broadcast Progress Update

Primary Actor: System (Automated)
Secondary Actor: TCP Client
Goal: Notify connected clients of progress changes
Preconditions: TCP server has active connections
Postconditions: All relevant clients receive update

Main Success Scenario: 1. System receives progress update from HTTP API

2. TCP server receives broadcast message via channel
3. Server identifies connections for the specific user
4. Server sends JSON progress message to connections
5. Clients receive and process update

Alternative Flows: - A1: Client connection lost - Server removes from active list

- A2: Send fails - Server logs error and continues with other clients

3.4 Notification System

UC-009: Register for UDP Notifications

Primary Actor: UDP Client
Goal: Register to receive chapter release notifications
Preconditions: UDP server is running
Postconditions: Client is registered for notifications

Main Success Scenario: 1. Client sends UDP registration packet with user preferences

2. Server receives registration and extracts client address
3. Server adds client to notification list
4. Server sends confirmation packet to client
5. Client is ready to receive notifications

UC-010: Send Chapter Release Notification

Primary Actor: System Administrator
Goal: Notify users about new chapter releases
Preconditions: UDP server has registered clients
Postconditions: Notification is broadcasted to clients

Main Success Scenario: 1. Administrator triggers notification for specific manga

2. System creates notification message with manga details

3. UDP server broadcasts message to all registered clients
4. Clients receive notification and display to users
5. System logs successful broadcast

Alternative Flows: - A1: Client unreachable - Server continues with other clients

- A2: Network error - Server logs error and retries

3.5 Real-time Chat System

UC-011: Join Chat

Primary Actor: Chat User
Goal: Connect to real-time chat system
Preconditions: User is authenticated, WebSocket server running
Postconditions: User is connected to chat

Main Success Scenario: 1. User's browser initiates WebSocket connection

2. Server upgrades HTTP connection to WebSocket
3. Client sends join message with user credentials
4. Server validates user and adds to active connections
5. Server broadcasts user join notification to other users
6. User receives recent chat history

UC-012: Send Chat Message

Primary Actor: Chat User
Goal: Send message to other connected users
Preconditions: User is connected to chat
Postconditions: Message is broadcasted to all users

Main Success Scenario: 1. User types message and clicks send

2. Client sends message via WebSocket connection
3. Server receives message and validates user
4. Server broadcasts message to all connected clients
5. All users receive and display the message

Alternative Flows: - A1: Message too long - Server returns error to sender

- A2: User not authenticated - Server rejects message

UC-013: Handle User Disconnection

Primary Actor: System (Automated)
Goal: Clean up when user leaves chat
Preconditions: User was connected to chat
Postconditions: User is removed from active connections

Main Success Scenario: 1. System detects WebSocket connection closure

2. Server removes connection from active list
3. Server broadcasts user leave notification
4. Other users see updated participant list
5. Connection resources are cleaned up

3.6 gRPC Internal Services

UC-014: Retrieve Manga via gRPC

Primary Actor: Internal Service
Goal: Fetch manga data through gRPC interface
Preconditions: gRPC server is running
Postconditions: Manga data is returned

Main Success Scenario: 1. Client service calls GetManga gRPC method

2. gRPC server receives request with manga ID
3. Server queries database for manga information
4. Server constructs protobuf response message
5. Server returns manga data to client

UC-015: Search Manga via gRPC

Primary Actor: Internal Service
Goal: Search manga using gRPC interface
Preconditions: gRPC server is running, database populated
Postconditions: Search results are returned

Main Success Scenario: 1. Client calls SearchManga with search criteria

2. gRPC server processes search parameters
3. Server executes database query with filters
4. Server constructs response with result list

5. Server returns paginated results to client

UC-016: Update Progress via gRPC

Primary Actor: Internal Service
Goal: Update user reading progress through gRPC
Preconditions: User and manga exist
Postconditions: Progress is updated in database

Main Success Scenario: 1. Client calls UpdateProgress with user and manga data

2. gRPC server validates request parameters
3. Server updates user_progress table
4. Server triggers TCP broadcast for real-time sync
5. Server returns success confirmation

4. Bonus Feature Use Cases

4.1 Enhanced Search and Filtering

UC-017: Advanced Manga Search

Primary Actor: Manga Reader
Goal: Search manga with multiple criteria
Preconditions: Advanced search feature is implemented
Postconditions: Filtered results are displayed

Main Success Scenario: 1. User opens advanced search interface

2. User selects genres, status, rating range, and year filters
3. System constructs complex database query
4. System applies full-text search on titles and descriptions
5. System returns ranked results based on relevance

4.2 User Reviews and Ratings

UC-018: Submit Manga Review

Primary Actor: Manga Reader
Goal: Write and publish manga review
Preconditions: User has manga in completed list
Postconditions: Review is published

Main Success Scenario: 1. User navigates to manga and clicks "Write Review"

2. User writes review text and assigns rating (1-10)
3. System validates review content and rating
4. System saves review to database with timestamp
5. System displays review on manga page

UC-019: View Manga Reviews

Primary Actor: Manga Reader
Goal: Read community reviews for manga
Preconditions: Manga has published reviews
Postconditions: Reviews are displayed

Main Success Scenario: 1. User views manga details page

2. System retrieves all reviews for the manga
3. System calculates average rating from all reviews
4. System displays reviews sorted by helpfulness or date
5. User can read individual reviews and ratings

4.3 Friend System

UC-020: Add Friend

Primary Actor: Manga Reader
Goal: Connect with another user as friend
Preconditions: Both users have accounts
Postconditions: Friend relationship is established

Main Success Scenario: 1. User searches for friend by username

2. User sends friend request
3. System notifies target user of friend request
4. Target user accepts friend request
5. System creates bidirectional friend relationship

UC-021: View Friend Activity

Primary Actor: Manga Reader
Goal: See reading activity of friends
Preconditions: User has friends
Postconditions: Activity feed is displayed

Main Success Scenario: 1. User accesses friends activity page

2. System retrieves recent activities from friends
3. System displays activities (completed manga, reviews, ratings)
4. Activities are sorted by recency
5. User can click through to view details

4.4 Reading Statistics

UC-022: Generate Reading Statistics

Primary Actor: System (Automated)
Goal: Calculate user's reading statistics
Preconditions: User has reading history
Postconditions: Statistics are computed and cached

Main Success Scenario: 1. System analyzes user's reading progress data

2. System calculates total chapters read, favorite genres
3. System determines reading patterns and trends
4. System generates monthly/yearly statistics
5. Statistics are cached for performance

UC-023: View Personal Statistics

Primary Actor: Manga Reader
Goal: View personal reading analytics
Preconditions: User has reading history
Postconditions: Statistics dashboard is displayed

Main Success Scenario: 1. User accesses statistics page

2. System retrieves cached statistics or generates new ones
3. System displays charts and graphs of reading activity
4. User can view different time periods and breakdowns
5. System shows reading goals progress if set

4.5 Caching and Performance

UC-024: Cache Popular Manga Data

Primary Actor: System (Automated)
Goal: Improve performance by caching frequently accessed data

Preconditions: Redis cache is available
Postconditions: Popular data is cached

Main Success Scenario: 1. System identifies frequently requested manga

2. System stores manga details in Redis cache
3. System sets appropriate cache expiration times
4. Subsequent requests serve data from cache
5. System updates cache when data changes

5. Error Handling and Recovery Use Cases

5.1 Database Connection Failure

UC-025: Handle Database Unavailability

Goal: Maintain partial functionality when database is unavailable
Trigger: Database connection fails

Success Criteria: - Read operations return cached data when available

- Write operations are queued for later processing
- Users receive appropriate error messages
- System attempts automatic reconnection

5.2 Network Service Failures

UC-026: TCP Server Recovery

Goal: Recover TCP service after failure
Trigger: TCP server crashes or network issues

Success Criteria: - Server automatically restarts and listens for connections

- Existing connections are gracefully handled
- Client reconnection is supported
- Progress updates are queued during downtime

UC-027: WebSocket Connection Recovery

Goal: Handle WebSocket connection interruptions
Trigger: Network connectivity issues

Success Criteria: - Client automatically attempts reconnection

- Chat history is preserved during disconnection
- User rejoins chat seamlessly after reconnection
- Other users are notified of connection status

6. Performance and Scalability Use Cases

6.1 Load Handling

UC-028: Support Concurrent Users

Goal: Handle multiple simultaneous users effectively
Trigger: 50-100 concurrent users access system

Success Criteria: - API response times remain under 500ms

- Database queries complete efficiently
- TCP and WebSocket connections remain stable
- No data corruption under concurrent access

UC-029: Efficient Data Retrieval

Goal: Optimize database queries for performance
Trigger: Large dataset queries or high request volume

Success Criteria: - Search queries complete within acceptable time limits

- Pagination prevents memory issues
- Indexes improve query performance
- Connection pooling manages database resources

7. Security Use Cases

7.1 Authentication and Authorization

UC-030: Validate JWT Tokens

Goal: Ensure only authenticated users access protected resources
Trigger: Request to protected endpoint

Success Criteria: - Invalid tokens are rejected

- Expired tokens trigger reauthentication
- Token claims are properly validated

- Unauthorized access is prevented

UC-031: Input Validation

Goal: Prevent malicious input from compromising system
Trigger: User input received

Success Criteria: - SQL injection attempts are blocked

- XSS attempts are sanitized
- Input length limits are enforced
- Invalid data formats are rejected

8. Success Metrics and Acceptance Criteria

8.1 Core Functionality Metrics

- **User Registration**: 100% of valid registrations succeed
- **Authentication**: <100ms token generation time
- **Manga Search**: Results returned within 500ms for 90% of queries
- **Progress Sync**: Updates broadcasted within 1 second
- **Chat Messages**: Real-time delivery within 100ms

8.2 Reliability Metrics

- **System Uptime**: 90% availability during testing period
- **Error Handling**: All error conditions handled gracefully
- **Data Consistency**: 100% accuracy in progress tracking
- **Connection Management**: Proper cleanup of all network connections

8.3 Performance Metrics

- **Concurrent Users**: Support for 50-100 simultaneous users
- **Database Operations**: <200ms for simple queries
- **Memory Usage**: Stable memory consumption under load
- **Network Protocol Efficiency**: Minimal bandwidth usage for sync operations

9. Implementation Priority (10-12 week)

9.1 Phase 1 (Weeks 1-3): Core HTTP Functionality

- User registration and authentication
- Basic manga CRUD operations
- Simple search functionality
- Database integration

9.2 Phase 2 (Weeks 4-8): Network Protocols

- TCP progress synchronization
- UDP notification system
- WebSocket real-time chat
- gRPC internal services

9.3 Phase 3 (Weeks 9-11): Integration and Polish

- End-to-end system integration
- Error handling and recovery
- Performance optimization
- Basic testing

9.4 Phase 4 (Week 12): Demo and Documentation

- Live demonstration preparation
- Technical documentation completion
- Final system validation

This use case specification provides a comprehensive but realistic foundation for implementing MangaHub within the constraints of a 10-12-week academic project while maintaining educational value and practical learning outcomes.