By: Khaden Watson

TV Show Progress Tracker - Product Requirements Document

1. Executive Summary

1.1 Product Overview

The TV Show Progress Tracker is a Java-based console application that allows users to manage and track their television viewing progress. Users can categorize shows into three states: "Plan to Watch," "Currently Watching," and "Finished Watching," providing a personal entertainment management system.

1.2 Target Audience

- Individual TV show enthusiasts who want to organize their viewing habits
- Users who consume multiple shows simultaneously and need progress tracking
- Entertainment enjoyers who want to maintain viewing records

1.3 Business Objectives

- Provide a simple, efficient way for users to track TV show progress
- Demonstrate proficiency in Java development, database integration, and software engineering best practices
- Create a scalable foundation for potential future enhancements

2. Product Vision and Goals

2.1 Vision Statement

To create an intuitive, reliable application that helps users organize and track their TV show viewing progress while maintaining complete control over their personal data.

2.2 Success Metrics

- Successful user authentication and session management

- Accurate progress tracking across multiple shows
- Data persistence and integrity
- Intuitive user navigation through console interface

3. Functional Requirements

3.1 Core Features

3.1.1 User Authentication

- **FR-1**: Users must log in with username and password
- **FR-2**: System must validate user credentials against database
- **FR-3**: Users can only access their own tracker data
- **FR-4**: Failed login attempts must be handled

3.1.2 TV Show Management

- **FR-5**: System must provide at least 10 TV shows for users to choose from
- **FR-6**: Users can add shows to their personal tracker
- **FR-7**: Users can categorize shows into three states:
    - Plan to Watch (Not Completed)
    - Currently Watching (In Progress)
    - Finished Watching (Completed)

3.1.3 Progress Tracking

- **FR-8**: Users can view all shows in their tracker organized by status
- **FR-9**: Users can update show status (move between categories)
- **FR-10**: Users can remove shows from their tracker
- **FR-11**: System must maintain history of user changes

3.1.4 Console Interface

- **FR-12**: Application must provide a menu-driven console interface
- **FR-13**: Navigation options must include:
    - View tracker by status
    - Add new show to tracker
    - Update show status
    - Remove show from tracker
    - Logout

3.2 Extended Features

3.2.1 Advanced Progress Tracking

- **FR-14**: For shows in progress, users can specify episode progress (e.g., "12 out of 50 episodes")
- **FR-15**: System calculates completion percentage for in-progress shows

3.2.2 User Management

- **FR-16**: New users can create accounts
- **FR-17**: Account creation requires unique username validation

3.2.3 Rating System

- **FR-18**: Users can rate completed shows (1-5 stars)
- **FR-19**: System calculates and displays average ratings per show

3.2.4 Administrative Features

- **FR-20**: Admin users can add new TV shows to the system
- **FR-21**: Admin users can edit existing show information
- **FR-22**: Admin users can remove shows from the system

3.2.5 Reporting

- **FR-23**: Users can view tracker reports showing:
    - Total shows in each category
    - Completion statistics
- **FR-24**: System provides aggregated reports showing:
    - How many users are watching each show
    - Completion rates per show

3.2.6 User Interface Enhancement

- **FR-25**: Optional simple UI to display tracker status visually

4. Technical Requirements

4.1 Database Requirements

- **TR-1**: Data must be stored in MySQL database
- **TR-2**: Minimum required tables:
    - Users table
    - Trackers table
- **TR-3**: Database must be initialized with SQL script containing sample data

4.2 Architecture Requirements

- **TR-4**: Application must be structured as Maven project
- **TR-5**: Database connectivity must use JDBC with DAO pattern
- **TR-6**: Must implement at least 2 custom exceptions:
    - DatabaseException
    - TrackerNotFoundException

4.3 Development Requirements

- **TR-7**: Project must use Git for version control
- **TR-8**: Development must follow Agile principles
- **TR-9**: Must maintain Kanban board for task tracking
- **TR-10**: Code must include proper error handling

4.4 Data Model Requirements

- **TR-11**: Must provide Entity-Relationship (ER) diagram
- **TR-12**: Database relationships must maintain referential integrity
- **TR-13**: User passwords must be stored securely (hashed)

5. Non-Functional Requirements

5.1 Performance

- **NFR-1**: Application must respond to user input
- **NFR-2**: Database queries must execute efficiently
- **NFR-3**: System must handle user sessions

5.2 Security

- **NFR-4**: User authentication must be secure
- **NFR-5**: Users cannot access other users' data

5.3 Usability

- **NFR-6**: Console interface must be intuitive and user-friendly
- **NFR-7**: Error messages must be clear

5.4 Reliability

- **NFR-8**: Data must persist between application sessions
- **NFR-9**: System must handle database connection failures
- **NFR-10**: Application must not crash on invalid user input

6. User Stories

6.1 Core User Stories

Epic: User Authentication

- As a user, I want to log in with my username and password so that I can access my personal tracker
- As a user, I want my data to be secure so that only I can access my viewing progress

Epic: Show Management

- As a user, I want to see a list of available TV shows so that I can choose what to track
- As a user, I want to add shows to my tracker so that I can organize my viewing
- As a user, I want to categorize shows by my viewing status so that I can see what I plan to watch, am currently watching, and have completed

Epic: Progress Tracking

- As a user, I want to update my show status so that my tracker reflects my current viewing progress
- As a user, I want to see all my tracked shows organized by status so that I can quickly understand my viewing situation
- As a user, I want to remove shows from my tracker so that I can maintain a clean, relevant list

6.2 Extended User Stories

Epic: Account Management

- As a new user, I want to create an account so that I can start tracking my shows
- As a user, I want to see my viewing statistics so that I can understand my viewing habits

Epic: Administrative Functions

- As an admin, I want to add new shows to the system so that users have more options to track
- As an admin, I want to see system-wide statistics so that I can understand user engagement

7. Acceptance Criteria

7.1 Core Features Acceptance Criteria

User Authentication

- GIVEN a valid username and password, WHEN I attempt to log in, THEN I should be authenticated and see my personal tracker
- GIVEN invalid credentials, WHEN I attempt to log in, THEN I should see an appropriate error message
- GIVEN I am logged in, WHEN I access tracker data, THEN I should only see my own data

Show Management

- GIVEN I am logged in, WHEN I view available shows, THEN I should see at least 10 different TV shows
- GIVEN I select a show, WHEN I add it to my tracker, THEN it should appear in my "Plan to Watch" category
- GIVEN I have shows in my tracker, WHEN I view my tracker, THEN shows should be organized by status

Progress Updates

- GIVEN I have a show in "Plan to Watch," WHEN I move it to "Currently Watching," THEN it should appear in the correct category
- GIVEN I have a show in "Currently Watching," WHEN I mark it as "Finished," THEN it should move to the completed category
- GIVEN I have shows in my tracker, WHEN I choose to remove one, THEN it should be deleted from my tracker

8. Technical Specifications

8.1 Database Schema

SQL

```sql
-- Users Table
CREATE TABLE User (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    is_admin BOOLEAN DEFAULT FALSE
);

-- Tracker Table
```

```sql
CREATE TABLE Tracker (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    show_id INT,
    episodes_watched INT DEFAULT 0,
    status VARCHAR(20),
    rating INT,
    FOREIGN KEY (user_id) REFERENCES User(id),
    FOREIGN KEY (show_id) REFERENCES Show(id)
);
```

```sql
--Show Table
CREATE TABLE Show (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    total_episodes INT NOT NULL
);
```

8.2 Application Architecture

Package Structure:

```
None
src/main/java/com/capstone/tvshowtracker/
├── App.java
├── dao/
│   ├── UserDAO.java
│   ├── ShowDAO.java
│   └── TrackerDAO.java
├── model/
│   ├── User.java
```

```
|    ├── TVShow.java
|    └── UserShowTracker.java
      |____ Show.java
├── service/
|    ├── UserService.java
|    ├── TrackerService.java
|    └── ShowService.java
├── exception/
|    ├── AuthenticationException.java
|    └── DataAccessException.java
└── util/
     └── DBUtil.java
```

9. Risk Assessment

9.1 Technical Risks

- **Database connectivity issues**: Mitigated by proper connection pooling and error handling
- **Data integrity concerns**: Mitigated by proper foreign key constraints and validation
- **Performance with large datasets**: Mitigated by efficient queries and indexing

9.2 Project Risks

- **Scope creep**: Mitigated by clear prioritization of core vs. extended features
- **Time constraints**: Mitigated by phased development approach
- **Complexity management**: Mitigated by modular architecture and clear separation of concerns

10. Success Criteria

The project will be considered successful when:

- All core functional requirements are implemented and tested
- User authentication works securely

- Data persists correctly across sessions
- Console interface is intuitive and bug-free
- Code demonstrates proper software engineering practices
- Database design follows normalization principles

11. Future Enhancements

Potential future improvements include:

- Web-based interface
- Advanced analytics and insights