Software Requirements Specification (SRS)

GoodReads Clone (PHP Backend)

Version 1.0

1. Introduction

1.1 Purpose

This document outlines the requirements for developing a GoodReads-inspired book review platform using pure PHP and MySQL. It defines core features, constraints, and milestones to ensure the team delivers an MVP without scope creep.

1.2 Document Conventions

- Follows IEEE SRS structure: numbered sections, functional/non-functional requirements, data models, and API endpoints.
- Code blocks use monospace formatting.
- Tables define database schemas and milestones.

1.3 Intended Audience

- Developers: PHP backend team (Team **Code Worriors**).
- Stakeholders: Course Instructor/Project mentor, team members.

1.4 Scope

In-Scope Features:

- User authentication (registration/login, profiles).
- Book management (browse/search, shelves).
- Reviews, ratings, and comments.
- Social features (follow users, activity feed, like, comment).

Out-of-Scope:

- Purchasing books.
- Advanced recommendation algorithms.

2. Overall Description

2.1 Product Perspective

A standalone web application with:

- Frontend: Prebuilt pages (HTML/CSS/JavaScript).
- Backend: Pure PHP with MySQL database.

2.2 Product Functions

Feature	Description
User Authentication	Secure registration/login with sessions.
Book Management	Admins add/edit/delete books; users search.
Shelves	Track books as "to-read," "reading," "read."

Reviews & Ratings	Submit/delete reviews (1–5 stars).
Social Interaction	Follow users, comment on reviews.

2.3 User Characteristics

- Role-based Authentication and Authorization

User Type	Description
Readers / Users	Browse books, write reviews, and manage shelves. Comfortable with web UIs.
Admins	Manage book catalog (CRUD operations). Moderate content.

2.4 Constraints

- Backend: Pure PHP (no frameworks).
- Database: MySQL or MariaDB backend.
- Timeline: Deliver MVP in ~4 weeks.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Authentication

- Users register with email, username, and password.
- Passwords stored using password_hash().
- Session management for login/logout.

- Register (POST /api/auth/register.php): email, username, password (hashed).
- Login (POST /api/auth/login.php): username/email + password → session/token.
- Logout (POST /api/auth/logout.php): invalidate session.

3.1.2 Book Catalog Management

- Admins add books with title, author, genre, and description.
- Users search for books by title/author/genre.

- List Books (GET /api/books/list.php): paginated.
- Book Details (GET /api/books/detail.php?id={book_id}): includes title, author, description, average rating, cover image.
- Search (GET /api/books/search.php?q={query}): by title/author.

3.1.3 Bookshelves

- Default Shelves: “to-read”, “currently-reading”, “read”.
- Custom Shelves: user-created (“Favorites”, “Did Not Finish”, etc.).
- Endpoints: add/remove book to shelf, list user’s shelves and contents.

3.1.4 Ratings & Reviews

- Submit reviews with star ratings (1–5).
- Delete own reviews.

- Submit Rating (POST /api/reviews/rate.php): 1–5 stars.
- Submit Review (POST /api/reviews/comment.php): text review.
- Fetch Reviews (GET /api/reviews/list.php?book_id={book_id}).

3.1.5 Social / Follow

- Follow/unfollow users.
- View the activity feed of followed users.
- View follower and following counts.
- Follower Count (GET /api/social/followers_count.php?user_id={target}).
- Following Count (GET /api/social/following_count.php?user_id={target}).
- Follow User (POST /api/social/follow.php?user_id={target}).
- Unfollow User (POST /api/social/unfollow.php?user_id={target}).
- Activity Feed (GET /api/social/feed.php): recent shelf/rating/review actions by followed users.

3.2 Non-Functional Requirements

Requirement	Description
Security	Passwords are hashed (bcrypt); input validation to prevent against XSS and SQL injection prevention.
Performance	Page load <2s for 100 concurrent users.

Usability	Responsive UI compatible with the existing frontend. JSON responses with consistent status codes and error messages.

4. Database Schema and Data Models

Table: Roles

```
-- Roles Table
CREATE TABLE `roles` (
  `id` INT PRIMARY KEY AUTO_INCREMENT,
  `name` VARCHAR(50) NOT NULL UNIQUE,
  `description` TEXT,
  `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Table: users

```
-- Users Table

CREATE TABLE `users` (

  `id` INT PRIMARY KEY AUTO_INCREMENT,

  `name` VARCHAR(255) NOT NULL,

  `email` VARCHAR(255) UNIQUE NOT NULL,

  `password` VARCHAR(255) NOT NULL,

  `profile_pic` VARCHAR(255) DEFAULT 'default.png',
```

```
  `bio` TEXT,
  `role_id` INT NOT NULL DEFAULT 2, -- Default to regular user role
  `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (`role_id`) REFERENCES `roles`(`id`)
);
```

Table: books

```
-- Books Table
CREATE TABLE `books` (
  `id` INT PRIMARY KEY AUTO_INCREMENT,
  `title` VARCHAR(255) NOT NULL,
  `author` VARCHAR(255) NOT NULL,
  `genre` VARCHAR(100),
  `description` TEXT,
  `published_year` YEAR
);
```

Table: shelves

```
-- Shelves Table
CREATE TABLE `shelves` (
  `id` INT PRIMARY KEY AUTO_INCREMENT,
  `user_id` INT,
```

```
  `book_id` INT,
  `status` ENUM('to-read', 'reading', 'read') DEFAULT 'to-read',
  FOREIGN KEY (`user_id`) REFERENCES `users`(`id`),
  FOREIGN KEY (`book_id`) REFERENCES `books`(`id`)
);
```

Table: shelf_books

```
-- Shelves Table
CREATE TABLE `shelves` (
  `id` INT PRIMARY KEY AUTO_INCREMENT,
  `user_id` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  FOREIGN KEY (`user_id`) REFERENCES `users`(`id`)
);
```

Table: reviews

```
CREATE TABLE `reviews` (
  `id` INT PRIMARY KEY AUTO_INCREMENT,
  `user_id` INT NOT NULL,
  `book_id` INT NOT NULL,
  `rating` INT CHECK (`rating` BETWEEN 1 AND 5),
  `comment` TEXT,
```

```
  `comment_count` INT DEFAULT 0,
  `like_count` INT DEFAULT 0,
  `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (`user_id`) REFERENCES `users`(`id`),
  FOREIGN KEY (`book_id`) REFERENCES `books`(`id`)
);
```

Table: Comments

```
-- Comments Table
CREATE TABLE `comments` (
  `id` INT PRIMARY KEY AUTO_INCREMENT,
  `review_id` INT NOT NULL,
  `user_id` INT NOT NULL,
  `content` TEXT NOT NULL,
  `like_count` INT DEFAULT 0, -- Count of likes
  `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (`review_id`) REFERENCES `reviews`(`id`),
  FOREIGN KEY (`user_id`) REFERENCES `users`(`id`)
);
```

Table: follows

```
-- Follows Table

CREATE TABLE `comments` (

  `id` INT PRIMARY KEY AUTO_INCREMENT,

  `review_id` INT NOT NULL,

  `user_id` INT NOT NULL,

  `content` TEXT NOT NULL,

  `like_count` INT DEFAULT 0, -- Count of likes

  `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  FOREIGN KEY (`review_id`) REFERENCES `reviews`(`id`),

  FOREIGN KEY (`user_id`) REFERENCES `users`(`id`)

);
```

Table : Review Likes

```
-- Review Likes Table

CREATE TABLE `review_likes` (

  `id` INT PRIMARY KEY AUTO_INCREMENT,

  `review_id` INT NOT NULL,

  `user_id` INT NOT NULL,

  `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
  UNIQUE (`review_id`, `user_id`), -- Prevent duplicate likes
  FOREIGN KEY (`review_id`) REFERENCES `reviews`(`id`),
  FOREIGN KEY (`user_id`) REFERENCES `users`(`id`)
);
```

Table: Comment Likes

```
-- Comment Likes Table
CREATE TABLE `comment_likes` (
  `id` INT PRIMARY KEY AUTO_INCREMENT,
  `comment_id` INT NOT NULL,
  `user_id` INT NOT NULL,
  `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE (`comment_id`, `user_id`), -- Prevent duplicate likes
  FOREIGN KEY (`comment_id`) REFERENCES `comments`(`id`),
  FOREIGN KEY (`user_id`) REFERENCES `users`(`id`)
);
```

5. API Endpoints

Category	Endpoint	Description

Auth	/api/auth/register.php (*POST*)	Create new user
Auth	/api/auth/login.php (*POST*)	Obtain session token
Auth	/api/auth/logout.php (*POST*)	End session
Books	/api/books/list.php (*GET*)	Paginated book list
Books	/api/books/detail.php?id={book_id} (*GET*)	Book info + avg. rating + review count
Books	/api/books/search.php?q={query} (*GET*)	Search by title/author
Shelves	/api/shelves/list.php (*GET*)	User's shelves + contents
Shelves	/api/shelves/add.php (*POST*)	Add book to shelf
Shelves	/api/shelves/remove.php (*POST*)	Remove book from shelf
Reviews	/api/reviews/list.php?book_id={book_id} (*GET*)	All reviews for a book
Reviews	/api/reviews/submit.php (*POST*)	Submit or update rating/review
Reviews	/api/reviews/delete.php (*POST*)	Delete user's review

Comments	/api/comments/list.php?review_id={review_id} (*GET*)	List comments for a review
Comments	/api/comments/submit.php (*POST*)	Submit comment on a review
Comments	/api/comments/delete.php (*POST*)	Delete a comment
Likes	/api/likes/review.php (*POST*)	Like/unlike a review
Likes	/api/likes/comment.php (*POST*)	Like/unlike a comment
Social	/api/social/follow.php?user_id={target} (*POST*)	Follow a user
Social	/api/social/unfollow.php?user_id={target} (*POST*)	Unfollow a user
Social	/api/social/followers_count.php?user_id={target} (*GET*)	Get follower count
Social	/api/social/following_count.php?user_id={target} (*GET*)	Get following count
Social	/api/social/feed.php (*GET*)	Recent activity of followed users

6. Folder structure

```
/backend/
├── auth/
│   ├── register.php
│   ├── login.php
│   └── logout.php
├── books/
│   ├── list.php
│   ├── detail.php
│   ├── search.php
│   ├── add.php (admin)
│   ├── edit.php (admin)
│   └── delete.php (admin)
├── shelves/
│   ├── list.php
│   ├── add.php
│   └── remove.php
├── reviews/
│   ├── list.php
│   ├── submit.php
│   └── delete.php
├── comments/
│   ├── list.php
│   ├── submit.php
│   └── delete.php
├── likes/
│   ├── review.php
│   └── comment.php
├── social/
│   ├── follow.php
│   ├── unfollow.php
│   ├── followers_count.php
│   ├── following_count.php
│   └── feed.php
├── includes/
│   ├── db.php (database connection)
│   └── utils.php (helper functions)
└── index.php (homepage)
```

7. Milestones & Timeline

Day	Milestone	Deliverables
Day 1	Project Setup	Repo structure, database schema, db.php
Day 3	Authentication	Register/login/logout endpoints + frontend hooks
Day 5	Book Catalog & Search	List/detail/search endpoints + basic UI integration

Day 7	Shelving System	Default/custom shelves + add/remove endpoints
Day 9	Reviews & Ratings	Rating/comment endpoints + display in book detail page
Day 11	Social Features	Follow/unfollow, activity feed, follower/following counts
Day 13	Likes System	Like/unlike endpoints for reviews and comments
Day 14	Testing & Deployment	End-to-end tests, bug fixes, server setup

8. Appendices

8.1 Technology Stack

Backend: Pure PHP

Database: MySQL

Server: Apache

Libraries: PHPMailer (for password reset)