WriteShelf - Functional Documentation

Table of Contents

Overview

WriteShelf is a web platform for writers and readers to share and discover books. The application is built using Flask for the backend, MongoDB for the database, and vanilla JavaScript with modern HTML5/CSS3 for the frontend.

Backend Architecture

Technology Stack

- **Framework**: Flask (Python)
- **Database**: MongoDB
- **Authentication**: Flask-Session
- **File Storage**: Local file system for development

Core Components

1. Application Core (`app.py`)

- Handles route definitions
- Manages session-based authentication
- Implements core business logic
- Processes file uploads
- Manages database connections

2. Database Models

Users Collection

- Username (unique)
- Email
- Password (hashed)
- Profile information
- Preferences
- Following/Followers

Books Collection

- Title
- Author reference
- Cover image path
- Description
- Creation date
- Likes count
- Reviews count

Reviews Collection

- Book reference
- User reference
- Content
- Rating
- Timestamp

3. File Management

- Handles book cover uploads
- Manages user profile photos
- Implements file type validation
- Handles file storage and retrieval

Frontend Architecture

Technology Stack

- **HTML5**
- **CSS3**
- **Vanilla JavaScript**

- **Font Awesome** for icons

Core Components

1. Main Page (`main.html`)

- Book discovery feed
- Search functionality
- Navigation menu
- User authentication state
- Real-time search results

```
// Search Implementation
const searchInput = document.getElementById('searchInput');
searchInput.addEventListener('input', debounce(performSearch, 300));
```

2. Profile Page

- User information display
- Statistics (followers, following, books)
- Book list
- Follow/Unfollow functionality
- Profile editing

3. Writing Interface

- Book creation form
- Cover upload
- Rich text editing
- Auto-save functionality

4. Search Interface

- Real-time search results
- Filter options
- Result categorization
- Responsive grid layout

API Documentation

Authentication Endpoints

- **POST /api/login**
    - Authenticates user credentials
    - Returns session cookie
    - Handles invalid login attempts
- **POST /api/signup**
    - Creates new user accounts
    - Validates unique username/email
    - Sets up initial user profile
- **GET /api/logout**
    - Terminates user session
    - Clears session cookie

Book Endpoints

- **GET /api/books/search**
    - Searches books by title or author
    - Supports partial matching
    - Returns paginated results


```
{
  "query": "string",
  "results": [
    {
      "id": "string",
      "title": "string",
      "author": "string",
      "cover": "string",
      "likes": "integer",
      "reviews": "integer"
    }
  ]
}
```


User Endpoints

- **GET /api/user/stats/{username}**
    - Returns user statistics
    - Includes follower counts
    - Shows book and review counts

- **POST /api/users/follow**
    - Handles follow/unfollow actions
    - Updates follower counts
    - Returns updated statistics

Database Schema

Users Collection

```
{
  "_id": ObjectId,
  "username": String,
  "email": String,
  "password_hash": String,
  "name": String,
  "photo": String,
  "bio": String,
  "following": [ObjectId],
  "followers": [ObjectId],
  "preferences": {
    "theme": String,
    "email_notifications": Boolean
  },
  "created_at": DateTime
}
```

Books Collection

```
{
  "_id": ObjectId,
  "title": String,
  "author_id": ObjectId,
  "cover": String,
  "description": String,
  "likes": [ObjectId],
  "reviews": [ObjectId],
  "created_at": DateTime,
  "updated_at": DateTime
}
```

Authentication & Authorization

Session Management

- Uses Flask-Session for server-side sessions
- Session timeout: 24 hours
- Secure cookie handling

Security Measures

- Password hashing using bcrypt
- CSRF protection
- Input sanitization
- File upload validation

Features

1. Book Management

- Create new books
- Upload cover images
- Edit book details
- Delete books

2. Social Features

- Follow/Unfollow users
- Like books
- Write reviews
- View activity feed

3. Search Functionality

- Real-time search
- Filter by title/author
- Sort results
- Advanced filtering

4. User Profiles

- Customizable profiles

- Statistics tracking
- Activity history
- Preference management

Development Guidelines

Code Style

- PEP 8 for Python code
- ESLint for JavaScript
- BEM methodology for CSS

Testing

- Unit tests for API endpoints
- Integration tests for user flows
- End-to-end testing for critical paths

Deployment

- **Development**: Local environment
- **Production**: Ready for cloud deployment
- Environment-specific configurations

Error Handling

- **API Errors**: Consistent error format, detailed error messages, appropriate HTTP status codes
- **Client-Side Validation**: Form validation, file upload checks, input sanitization

Performance Considerations

- **Backend**: Database indexing, query optimization, caching strategies
- **Frontend**: Asset optimization, lazy loading, performance monitoring

---

Security Measures

- **Data Protection**: Input validation, XSS prevention, CSRF protection, secure file handling
- **Authentication**: Session management, password policies, rate limiting

---

Maintenance

- **Monitoring**: Error logging, performance metrics, user analytics
- **Backup**: Database backup strategy, file storage backup, recovery procedures