Bookify

Software Requirements Specification

Version: 1.0
Date: December 8, 2025
Project: Digital & Physical Book Marketplace Platform

Table of Contents

---

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) defines the requirements for **Bookify** — a digital and physical book marketplace focused on one-time purchases. The platform uses Cloudinary for chunked PDF uploads and MongoDB for data storage. Authentication uses JWT access and refresh tokens with role-based access control (RBAC).

1.2 System Overview

Bookify allows users to browse, purchase (both digital and physical books), and read content securely. Authors can upload and manage their books, while administrators moderate content and manage platform operations. The system delivers secure reading experiences via signed URLs and supports comprehensive e-commerce functionality.

1.3 Scope

In Scope:

- User authentication and authorization with JWT
- Digital and physical book marketplace
- Chunked PDF upload and preview generation
- Shopping cart and checkout system
- Digital library and online reader
- Review and rating system
- Advanced search capabilities
- Author and admin dashboards

Out of Scope:

- Subscription-based models
- Social networking features
- Third-party integrations beyond payment gateways
- Mobile native applications (initial release)

1.4 Definitions and Acronyms

- **SRS:** Software Requirements Specification
- **JWT:** JSON Web Token
- **RBAC:** Role-Based Access Control
- **API:** Application Programming Interface
- **CRUD:** Create, Read, Update, Delete
- **PDF:** Portable Document Format
- **ISBN:** International Standard Book Number

2. System Features Summary

Bookify provides a comprehensive set of features to support all stakeholders:

- **Authentication & Authorization:** JWT access tokens with refresh token rotation and RBAC
- **Cloud Storage Integration:** Cloudinary chunked upload with automatic preview generation
- **E-Commerce Functionality:** One-time purchases for digital and physical books
- **Shopping Experience:** Persistent shopping cart, secure checkout, invoice generation, and order management
- **Digital Library:** User library with secure online reader and reading progress tracking
- **Social Features:** Reviews, ratings, wishlist, and personalized recommendations
- **Search Capabilities:** Advanced full-text search powered by MongoDB
- **Content Management:** Author dashboard for book management and basic sales analytics
- **Administration:** Complete admin tools for user, book, and order management

---

3. User Roles & Permissions

3.1 Guest

Capabilities:

- Browse book catalog
- View book details and previews
- Search books
- View public reviews and ratings

Restrictions:

- Cannot purchase books
- Cannot access library or reader
- Cannot leave reviews or use wishlist

3.2 Registered User

Inherits: All Guest capabilities

Additional Capabilities:

- Purchase digital and physical books
- Manage shopping cart
- Access personal digital library
- Use online reader with progress tracking
- Leave reviews and ratings
- Manage wishlist
- View order history and invoices
- Update profile information

3.3 Author

Inherits: All Registered User capabilities

Additional Capabilities:

- Upload and manage books
- Update book metadata
- View sales statistics
- Manage book pricing
- Access author dashboard

Restrictions:

- Requires admin approval to publish books
- Cannot manage other authors' content

3.4 Administrator

Inherits: All system capabilities

Exclusive Capabilities:

- Manage all users (CRUD operations)
- Moderate and manage all books

- Manage all orders
- Approve/reject author applications
- Generate system reports
- Configure platform settings
- Access analytics dashboard

---

4. Functional Requirements

4.1 Authentication & Authorization

FR-AUTH-001: User Registration

- Users must register with email, password, and basic profile information
- Email verification required before full access
- Password must meet security requirements (minimum 8 characters, mixed case, numbers, special characters)

FR-AUTH-002: User Login

- Users authenticate with email and password
- System generates JWT access token (15-minute expiry)
- System generates refresh token (7-day expiry) stored as httpOnly cookie

FR-AUTH-003: Token Refresh

- Users can refresh access token using valid refresh token
- Refresh token rotation implemented for security
- Automatic token refresh on API calls when access token expires

FR-AUTH-004: User Logout

- Invalidate refresh token on logout
- Clear client-side tokens
- Log logout event for security audit

FR-AUTH-005: Role-Based Access Control

- System enforces role-based permissions on all protected routes
- Middleware validates user role before granting access
- Unauthorized access attempts logged

4.2 Book Management

FR-BOOK-001: Book Upload (Authors)

- Authors upload PDF files in chunks via Cloudinary
- Support for files up to 100MB
- Automatic preview generation (first 5-10 pages)
- Progress indicator during upload

FR-BOOK-002: Book Metadata Management

- Authors provide: title, description, ISBN, genre, price, author info
- Support for digital-only, physical-only, or both formats
- Optional cover image upload
- Tags and categories for organization

FR-BOOK-003: Book Listing (Admin)

- Admin reviews and approves/rejects submitted books
- Admin can edit any book metadata
- Admin can unpublish or delete books
- Rejection requires reason provided to author

FR-BOOK-004: Book Preview

- All users can preview sample pages (5-10 pages)
- Preview delivered via Cloudinary transformation
- No download option for preview

4.3 Shopping Cart & Checkout

FR-CART-001: Add to Cart

- Users can add books (digital/physical) to cart
- Mixed cart support (both formats in same cart)

- Cart persists across sessions
- Quantity management for physical books

FR-CART-002: Cart Management

- Users can view, update quantities, and remove items
- Real-time price calculation
- Apply promotional codes (if available)
- Display stock availability for physical books

FR-CART-003: Checkout Process

- Collect shipping address for physical books
- Integrate payment gateway (Stripe/PayPal)
- Generate order confirmation
- Send confirmation email

FR-CART-004: Invoice Generation

- Generate PDF invoice for each order
- Include order details, pricing, taxes, and shipping
- Store invoice for user access
- Support invoice download

4.4 Digital Library & Reader

FR-LIB-001: User Library

- Display all purchased digital books
- Filter and sort options (date, title, author)
- Show reading progress for each book
- Access to download receipts

FR-LIB-002: Online Reader

- Deliver books via signed Cloudinary URLs (30-minute expiry)
- Page navigation (next, previous, jump to page)
- Track and save reading progress

- Responsive reader interface

FR-LIB-003: Reading Progress

- Auto-save current page on navigation
- Display progress percentage
- Resume from last read position
- Sync across devices (if logged in)

FR-LIB-004: Download Management

- Users can download purchased books (limited downloads)
- Track download count per book
- Watermark PDFs with user information

4.5 Reviews & Ratings

FR-REV-001: Leave Review

- Users can review purchased books only
- Rating (1-5 stars) required
- Optional text review (max 1000 characters)
- One review per user per book

FR-REV-002: Manage Reviews

- Users can edit/delete their own reviews
- Admin can moderate/remove inappropriate reviews
- Display average rating and review count on book page

FR-REV-003: Review Helpfulness

- Users can mark reviews as helpful
- Sort reviews by helpfulness, date, or rating

4.6 Wishlist

FR-WISH-001: Wishlist Management

- Users can add/remove books from wishlist
- Wishlist persists across sessions

- Display wishlist on user profile
- Notification when wishlisted book goes on sale

4.7 Search & Recommendations

FR-SRCH-001: Advanced Search

- Full-text search using MongoDB text indexes
- Search by title, author, ISBN, genre, tags
- Filter results by price, format, rating, publication date
- Sort results by relevance, price, rating, date

FR-SRCH-002: Autocomplete

- Real-time search suggestions as user types
- Show top 5-10 matching results
- Include book covers in suggestions

FR-SRCH-003: Recommendations

- Rule-based recommendations on book detail pages
- Based on genre, author, and user purchase history
- Display "Customers also bought" section
- Personalized recommendations on homepage (for logged-in users)

4.8 Author Dashboard

FR-AUTH-001: Book Management

- View all uploaded books and their status
- Edit book metadata for unpublished books
- Delete unpublished books
- View rejection reasons

FR-AUTH-002: Sales Analytics

- View total sales count per book
- View revenue per book
- Basic charts (sales over time)

- Export sales data as CSV

4.9 Admin Dashboard

FR-ADM-001: User Management

- View all users with role filters
- Edit user roles and status
- Deactivate/reactivate user accounts
- View user activity logs

FR-ADM-002: Book Management

- View all books with status filters
- Approve/reject pending books
- Edit any book metadata
- Unpublish/delete books

FR-ADM-003: Order Management

- View all orders with status filters
- Update order status (processing, shipped, delivered)
- View order details and customer information
- Generate shipping labels (future enhancement)

FR-ADM-004: Reporting

- Generate sales reports (daily, weekly, monthly)
- User registration and activity reports
- Revenue reports by book, author, genre
- Export reports as PDF/CSV

---

5. Non-Functional Requirements

5.1 Performance

NFR-PERF-001: Response Time

- API endpoints respond within 200ms for 95% of requests
- Search results display within 500ms
- Reader loads within 1 second

NFR-PERF-002: Scalability

- Support 10,000 concurrent users
- Handle 100 book uploads per hour
- Scale horizontally via containerization

NFR-PERF-003: Throughput

- Process 1,000 transactions per hour
- Support 50MB/s aggregate upload bandwidth

5.2 Security

NFR-SEC-001: Data Encryption

- All data in transit encrypted via HTTPS/TLS 1.3
- Passwords hashed using bcrypt (cost factor 12)
- Sensitive database fields encrypted at rest

NFR-SEC-002: Authentication Security

- JWT tokens signed with RS256 algorithm
- Refresh tokens use secure random generation
- Implement rate limiting on authentication endpoints (5 attempts per 15 minutes)

NFR-SEC-003: API Security

- Implement CORS with whitelist
- Rate limiting on all API endpoints (100 requests per minute per IP)
- Input validation and sanitization on all endpoints

NFR-SEC-004: File Security

- Validate file types and sizes on upload
- Scan uploaded files for malware
- Use signed URLs with expiration for content delivery

5.3 Reliability

NFR-REL-001: Availability

- System uptime of 99.9% (excluding scheduled maintenance)
- Maximum 1 hour planned downtime per month
- Graceful degradation during partial outages

NFR-REL-002: Data Integrity

- Database transactions ensure ACID properties
- Regular automated backups (daily full, hourly incremental)
- Point-in-time recovery capability

NFR-REL-003: Error Handling

- Graceful error messages for users
- Comprehensive error logging for debugging
- Automatic retry logic for transient failures

5.4 Usability

NFR-USE-001: User Interface

- Responsive design supporting desktop, tablet, mobile
- WCAG 2.1 AA accessibility compliance
- Intuitive navigation with max 3 clicks to any feature

NFR-USE-002: User Experience

- Consistent UI/UX across all pages
- Loading indicators for long operations
- Clear error messages with actionable guidance

5.5 Maintainability

NFR-MAIN-001: Code Quality

- Modular architecture with clear separation of concerns
- Code coverage minimum 80% for critical paths
- Comprehensive API documentation using OpenAPI/Swagger

NFR-MAIN-002: Monitoring

- Application performance monitoring (APM)
- Real-time error tracking and alerting
- System health dashboard

5.6 Compliance

NFR-COMP-001: Legal Compliance

- GDPR compliance for EU users
- PCI DSS compliance for payment processing
- Copyright and DMCA compliance mechanisms

NFR-COMP-002: Privacy

- Clear privacy policy and terms of service
- User consent management for data collection
- Right to deletion and data export

---

6. Data Model (Detailed Schema)

6.1 User Collection

```javascript
```

```
{
  _id: ObjectId,
  email: String (unique, required),
  password: String (hashed, required),
  firstName: String (required),
  lastName: String (required),
  role: String (enum: ['guest', 'user', 'author', 'admin'], default: 'user'),
  isEmailVerified: Boolean (default: false),
  profile: {
    avatar: String (URL),
    bio: String,
    phone: String,
    addresses: [{
      type: String (enum: ['shipping', 'billing']),
      street: String,
      city: String,
      state: String,
      zipCode: String,
      country: String,
      isDefault: Boolean
    }]
  },
  authorInfo: {
    isApproved: Boolean (default: false),
    approvedAt: Date,
    approvedBy: ObjectId (ref: User),
    rejectionReason: String,
    biography: String,
    website: String
  },
  refreshTokens: [{
    token: String (hashed),
    expiresAt: Date,
    createdAt: Date
  }],
  createdAt: Date (default: Date.now),
  updatedAt: Date,
  lastLoginAt: Date,
  isActive: Boolean (default: true)
}
```

6.2 Book Collection

javascript

```
{
  _id: ObjectId,
  isbn: String (unique, indexed),
  title: String (required, indexed),
  description: String,
  author: {
    userId: ObjectId (ref: User, required),
    name: String (required),
    bio: String
  },
  coverImage: {
    url: String,
    publicId: String (Cloudinary)
  },
  pdfFile: {
    url: String (Cloudinary),
    publicId: String,
    sizeInBytes: Number,
    pageCount: Number
  },
  preview: {
    url: String (Cloudinary signed),
    pageRange: String (e.g., "1-10")
  },
  pricing: {
    digital: {
      price: Number,
      currency: String (default: 'USD'),
      isAvailable: Boolean (default: true)
    },
    physical: {
      price: Number,
      currency: String (default: 'USD'),
      isAvailable: Boolean (default: false),
      stock: Number (default: 0)
    }
  },
  metadata: {
    genre: [String] (indexed),
    tags: [String],
    language: String (default: 'en'),
    publisher: String,
    publishedDate: Date,
```

```
  edition: String,
  pageCount: Number
 },
 stats: {
  totalSales: Number (default: 0),
  averageRating: Number (default: 0),
  reviewCount: Number (default: 0),
  viewCount: Number (default: 0)
 },
 status: String (enum: ['draft', 'pending', 'approved', 'rejected', 'unpublished'], default: 'draft'),
 adminReview: {
  reviewedBy: ObjectId (ref: User),
  reviewedAt: Date,
  rejectionReason: String
 },
 createdAt: Date (default: Date.now),
 updatedAt: Date,
 publishedAt: Date
}
```

6.3 Order Collection

javascript

```
{
  _id: ObjectId,
  orderNumber: String (unique, indexed),
  userId: ObjectId (ref: User, required),
  items: [{
    bookId: ObjectId (ref: Book, required),
    title: String,
    format: String (enum: ['digital', 'physical']),
    price: Number,
    quantity: Number (default: 1)
  }],
  pricing: {
    subtotal: Number,
    tax: Number,
    shipping: Number,
    discount: Number (default: 0),
    total: Number
  },
  shippingAddress: {
    street: String,
    city: String,
    state: String,
    zipCode: String,
    country: String
  },
  payment: {
    method: String (enum: ['stripe', 'paypal']),
    transactionId: String,
    status: String (enum: ['pending', 'completed', 'failed', 'refunded']),
    paidAt: Date
  },
  fulfillment: {
    status: String (enum: ['pending', 'processing', 'shipped', 'delivered', 'cancelled']),
    shippedAt: Date,
    deliveredAt: Date,
    trackingNumber: String,
    carrier: String
  },
  invoice: {
    url: String,
    generatedAt: Date
  },
  createdAt: Date (default: Date.now),
```

```
  updatedAt: Date
}
```

6.4 Review Collection

```javascript
{
  _id: ObjectId,
  bookId: ObjectId (ref: Book, required, indexed),
  userId: ObjectId (ref: User, required),
  rating: Number (required, min: 1, max: 5),
  reviewText: String (maxLength: 1000),
  helpfulCount: Number (default: 0),
  helpfulBy: [ObjectId] (ref: User),
  isVerifiedPurchase: Boolean (default: false),
  status: String (enum: ['active', 'hidden', 'deleted'], default: 'active'),
  moderatedBy: ObjectId (ref: User),
  moderatedAt: Date,
  createdAt: Date (default: Date.now),
  updatedAt: Date
}
```

6.5 Cart Collection

```javascript
{
  _id: ObjectId,
  userId: ObjectId (ref: User, unique, required),
  items: [{
    bookId: ObjectId (ref: Book, required),
    format: String (enum: ['digital', 'physical']),
    quantity: Number (default: 1),
    price: Number,
    addedAt: Date (default: Date.now)
  }],
  updatedAt: Date (default: Date.now)
}
```

6.6 Wishlist Collection

```javascript
```

```
{
  _id: ObjectId,
  userId: ObjectId (ref: User, unique, required),
  books: [{
    bookId: ObjectId (ref: Book, required),
    addedAt: Date (default: Date.now)
  }],
  updatedAt: Date (default: Date.now)
}
```

6.7 ReadingProgress Collection

javascript

```
{
  _id: ObjectId,
  userId: ObjectId (ref: User, required),
  bookId: ObjectId (ref: Book, required),
  currentPage: Number (default: 1),
  totalPages: Number,
  progressPercentage: Number (default: 0),
  lastReadAt: Date (default: Date.now),
  createdAt: Date (default: Date.now),
  updatedAt: Date
}
```

6.8 Download Collection

javascript

```
{
  _id: ObjectId,
  userId: ObjectId (ref: User, required),
  bookId: ObjectId (ref: Book, required),
  orderId: ObjectId (ref: Order, required),
  downloadCount: Number (default: 0),
  maxDownloads: Number (default: 5),
  lastDownloadAt: Date,
  createdAt: Date (default: Date.now)
}
```

---

7. API Endpoints (Detailed)

7.1 Authentication Endpoints

POST /api/auth/register

- **Description:** Register a new user
- **Request Body:**

```
json

{
  "email": "user@example.com",
  "password": "SecurePass123!",
  "firstName": "John",
  "lastName": "Doe"
}
```

- **Response:** 201 Created

```
json

{
  "success": true,
  "message": "User registered successfully. Please verify your email.",
  "data": {
    "userId": "64a1b2c3d4e5f6g7h8i9j0k1",
    "email": "user@example.com"
  }
}
```

POST /api/auth/login

- **Description:** Authenticate user and return tokens
- **Request Body:**

```
json

{
  "email": "user@example.com",
  "password": "SecurePass123!"
}
```

- **Response:** 200 OK

```json
{
  "success": true,
  "data": {
    "accessToken": "eyJhbGc...",
    "user": {
      "id": "64a1b2c3d4e5f6g7h8i9j0k1",
      "email": "user@example.com",
      "role": "user"
    }
  }
}
```

- **Note:** Refresh token set as httpOnly cookie

POST /api/auth/refresh

- **Description:** Refresh access token
- **Request:** Refresh token in httpOnly cookie
- **Response:** 200 OK

```json
{
  "success": true,
  "data": {
    "accessToken": "eyJhbGc..."
  }
}
```

POST /api/auth/logout

- **Description:** Logout user and invalidate refresh token
- **Headers:** Authorization: Bearer {accessToken}
- **Response:** 200 OK

```json
```

```
{
 "success": true,
 "message": "Logged out successfully"
}
```


7.2 Book Endpoints

GET /api/books

- **Description:** Get paginated list of books with filters
- **Query Parameters:**
  - `page` (default: 1)
  - `limit` (default: 20)
  - `genre` (optional)
  - `minPrice`, `maxPrice` (optional)
  - `format` (digital/physical)
  - `sort` (price_asc, price_desc, rating, newest)
- **Response:** `200 OK`


```
json
{
 "success": true,
 "data": {
  "books": [...],
  "pagination": {
   "currentPage": 1,
   "totalPages": 10,
   "totalBooks": 195,
   "limit": 20
  }
 }
}
```


GET /api/books/:id

- **Description:** Get detailed book information
- **Response:** `200 OK`

```json
{
 "success": true,
 "data": {
  "book": {
   "_id": "64a1b2c3d4e5f6g7h8i9j0k1",
   "title": "The Great Adventure",
   "author": {...},
   "pricing": {...},
   "preview": {...},
   "stats": {...}
  },
  "recommendations": [...]
 }
}
```

POST /api/books (Author only)

- **Description:** Create new book (chunked upload handled separately)
- **Headers:** `Authorization: Bearer {accessToken}`
- **Request Body:**

```json
{
 "isbn": "978-3-16-148410-0",
 "title": "My Book",
 "description": "...",
 "genre": ["Fiction", "Adventure"],
 "pricing": {
  "digital": { "price": 9.99 },
  "physical": { "price": 19.99, "stock": 100 }
 },
 "pdfFileId": "cloudinary_public_id"
}
```

- **Response:** `201 Created`

PUT /api/books/:id (Author/Admin)

- **Description:** Update book metadata

- **Headers:** `Authorization: Bearer {accessToken}`
- **Response:** `200 OK`

DELETE /api/books/:id (Author/Admin)

- **Description:** Delete book
- **Headers:** `Authorization: Bearer {accessToken}`
- **Response:** `204 No Content`

POST /api/books/:id/upload (Author only)

- **Description:** Initialize chunked upload session
- **Headers:** `Authorization: Bearer {accessToken}`
- **Response:** `200 OK` with upload session details

GET /api/books/:id/preview

- **Description:** Get preview signed URL
- **Response:** `200 OK`

```json
{
  "success": true,
  "data": {
    "previewUrl": "https://cloudinary.../signed_url",
    "expiresAt": "2025-12-08T15:30:00Z"
  }
}
```

7.3 Cart Endpoints

GET /api/cart (User only)

- **Description:** Get user's cart
- **Headers:** `Authorization: Bearer {accessToken}`
- **Response:** `200 OK`

POST /api/cart/items (User only)

- **Description:** Add item to cart

- **Request Body:**

```
json

{
  "bookId": "64a1b2c3d4e5f6g7h8i9j0k1",
  "format": "digital",
  "quantity": 1
}
```

- **Response:** `200 OK`

PUT /api/cart/items/:bookId (User only)

- **Description:** Update cart item quantity
- **Request Body:** `{ "quantity": 2 }`
- **Response:** `200 OK`

DELETE /api/cart/items/:bookId (User only)

- **Description:** Remove item from cart
- **Response:** `200 OK`

7.4 Order Endpoints

POST /api/orders/checkout (User only)

- **Description:** Create order from cart
- **Headers:** `Authorization: Bearer {accessToken}`
- **Request Body:**

```
json

{
  "shippingAddress": {...},
  "paymentMethod": "stripe",
  "paymentToken": "tok_xxx"
}
```

- **Response:** `201 Created`

GET /api/orders (User only)

- **Description:** Get user's order history
- **Headers:** `Authorization: Bearer {accessToken}`
- **Response:** `200 OK`

GET /api/orders/:id (User only)

- **Description:** Get order details
- **Headers:** `Authorization: Bearer {accessToken}`
- **Response:** `200 OK`

GET /api/orders/:id/invoice (User only)

- **Description:** Download invoice PDF
- **Headers:** `Authorization: Bearer {accessToken}`
- **Response:** PDF file

7.5 Library & Reader Endpoints

GET /api/library (User only)

- **Description:** Get user's digital library
- **Headers:** `Authorization: Bearer {accessToken}`
- **Response:** `200 OK`

GET /api/library/books/:bookId/reader (User only)

- **Description:** Get signed URL for reading
- **Headers:** `Authorization: Bearer {accessToken}`
- **Response:** `200 OK`

```
json
```

```
{
 "success": true,
 "data": {
  "readerUrl": "https://cloudinary.../signed_url",
  "expiresAt": "2025-12-08T15:30:00Z",
  "currentPage": 42,
  "totalPages": 350
 }
}
```


PUT /api/library/books/:bookId/progress (User only)

- **Description:** Update reading progress
- **Request Body:** `{ "currentPage": 45 }`
- **Response:** `200 OK`

POST /api/library/books/:bookId/download (User only)

- **Description:** Download purchased book
- **Headers:** `Authorization: Bearer {accessToken}`
- **Response:** Signed download URL

7.6 Review Endpoints

POST /api/books/:bookId/reviews (User only)

- **Description:** Create review for purchased book
- **Headers:** `Authorization: Bearer {accessToken}`
- **Request Body:**


```
json
{
 "rating": 5,
 "reviewText": "Amazing book!"
}
```


- **Response:** `201 Created`

GET /api/books/:bookId/reviews

- **Description:** Get book reviews (paginated)

- **Query Parameters:** `page`, `limit`, `sort` (helpful, recent, rating)
- **Response:** `200 OK`

PUT /api/reviews/:id (User - own review only)

- **Description:** Update own review
- **Response:** `200 OK`

DELETE /api/reviews/:id (User/Admin)

- **Description:** Delete review
- **Response:** `204 No Content`

POST /api/reviews/:id/helpful (User only)

- **Description:** Mark review as helpful
- **Response:** `200 OK`

7.7 Wishlist Endpoints

GET /api/wishlist (User only)

- **Description:** Get user's wishlist
- **Headers:** `Authorization: Bearer {accessToken}`
- **Response:** `200 OK`

POST /api/wishlist/items (User only)

- **Description:** Add book to wishlist
- **Request Body:** `{ "bookId": "..." }`
- **Response:** `200 OK`

DELETE /api/wishlist/items/:bookId (User only)

- **Description:** Remove from wishlist
- **Response:** `200 OK`

7.8 Search Endpoints

GET /api/search

- **Description:** Search books with full-text search

- **Query Parameters:**
  - `q` (search query, required)
  - `filters` (genre, price range, format)
  - `page`, `limit`
  - `sort`
- **Response:** `200 OK`

GET /api/search/autocomplete

- **Description:** Get search suggestions
- **Query Parameters:** `q` (partial query)
- **Response:** `200 OK`

7.9 Author Dashboard Endpoints

GET /api/author/books (Author only)

- **Description:** Get author's books
- **Headers:** `Authorization: Bearer {accessToken}`
- **Response:** `200 OK`

GET /api/author/sales (Author only)

- **Description:** Get sales analytics
- **Query Parameters:** `startDate`, `endDate`
- **Response:** `200 OK`

```json
{
  "success": true,
  "data": {
    "totalSales": 1250,
    "totalRevenue": 12450.00,
    "bookStats": [...]
  }
}
```

7.10 Admin Endpoints

GET /api/admin/users (Admin only)