**BookNest Backend Technical Documentation**

**Overview**

The **BookNest** backend is a robust and scalable server-side application designed to support an online book marketplace. Built with the **MERN** stack (specifically focusing on the Node.js + Express + MongoDB part), this system handles user management, book listings, cart operations, order processing, invoice generation, OTP-based authentication, and admin functionalities.

**Architecture**

**Layered Architecture**

* **Controller Layer**: Handles HTTP requests and delegates logic to services.
* **Service Layer**: Contains business logic and data handling operations.
* **Model Layer**: Defines Mongoose schemas and relationships.
* **Middleware Layer**: Manages authentication, authorization, and rate-limiting.
* **Utility Layer**: Handles logging, email, PDF generation, and other shared functionalities.

**Folder Structure**

├── config/ # Environment configs (e.g., mail)

├── controllers/ # Route handlers

├── middlewares/ # Auth, roles, and rate-limiters

├── models/ # Mongoose schemas

├── routes/ # Express route definitions

├── services/ # Core business logic

├── templates/ # Email and invoice templates

├── utils/ # Logging, email, pdf utilities

├── validators/ # Input validation logic

└── index.js # Application entry point

**Technologies Used**

|  |  |
| --- | --- |
| **Technology** | **Purpose** |
| Node.js + Express | Backend server and routing |
| MongoDB + Mongoose | NoSQL database and ODM |
| JWT | Authentication |
| bcryptjs | Password hashing |
| nodemailer | Email transmission |
| express-validator | Input validation |
| multer | File uploads |
| pdf-creator-node | PDF invoice generation |
| dotenv | Environment configuration |
| express-rate-limit | DDoS and brute-force protection |
| sanitize-html | Template sanitization |
| winston | Logging |

**Core Features**

* **User Management**
* Registration with OTP verification
* Login with JWT authentication
* Password reset via OTP
* **Book Module**
* Book listing, creation, update, and deletion
* Image upload support with multer
* **Cart Module**
* Add/update/remove book items
* Quantity-based cart operations
* Cart validation against stock (future enhancement)
* **Order Module**
* Create orders from cart
* Address and contact collection
* Order status management (Pending, Shipped, etc.)
* **Invoice Generation**
* Dynamic invoice PDF via pdf-creator-node
* Email attachment via nodemailer
* **Admin Features**
* Role-based access control using authorizeRoles()
* Route protection via authenticate and authorizeAdmin
* **Security & Validation**
* OTP throttling and expiry checks
* Express-validator for payload validation
* Rate limiting for auth and OTP endpoints

**Workflows**

* **Authentication Flow**

1. POST /otp/send → generates & emails OTP
2. POST /otp/verify → verifies OTP
3. POST /otp/register → creates user post-OTP
4. POST /auth/login → authenticates and returns JWT

* **Invoice Flow**

1. GET /invoice/:orderId → returns PDF invoice
2. POST /invoice/email → sends invoice via email

* **Cart & Order Flow**

1. POST /cart → adds item
2. PUT /cart → updates item quantity
3. POST /order → creates order

**Scalability and Modularity**

**Modularity**

* Logical separation of concerns (routes, services, models)
* Validators and utilities can be reused across modules
* Role middleware allows dynamic permissioning

**Scalability**

* Stateless JWT-based authentication
* MongoDB’s document model for flexible data growth
* Modular folder and route design enable microservice transition

**Testing (Future Enhancement)**

* Use **Jest** for unit and integration testing
* Coverage for:
  + OTP generation and validation
  + Order lifecycle
  + Cart mutation logic

**Security Highlights**

* Passwords hashed with bcryptjs
* Environment variables managed via dotenv
* Public endpoints protected with express-rate-limit
* Token expiry checks and error handling in authenticate
* Data sanitization with sanitize-html

**Package Overview**

"dependencies": {

"bcryptjs": "^3.0.2",

"body-parser": "^2.2.0",

"cors": "^2.8.5",

"dotenv": "^16.5.0",

"express": "^5.1.0",

"express-validator": "^7.2.1",

"jsonwebtoken": "^9.0.2",

"mongoose": "^8.15.2",

"multer": "^2.0.1",

"nodemailer": "^7.0.3",

"pdf-creator-node": "^2.3.5"

}

**Future Improvements**

* Stock validation before cart/order submission
* Admin dashboard APIs
* Test automation suite
* Audit logging for sensitive operations
* Internationalization (i18n) support
* Payment gateway integration (Razorpay)

**Contributors & Author**

* Pappu Kumar
* MD. Zamir Alam

Last reviewed and updated: **June 2025**