**Library Management System Documentation**

**1. System Overview**

The Library Management System is a modern, distributed application designed to manage library operations including book inventory, user management, and rental tracking. The system follows a microservices architecture pattern, consisting of three main components:

* Book Service: Manages book inventory and availability
* User Service: Handles user management and rental operations
* Frontend Application: Provides a user-friendly web interface

**Key Features**

* Book inventory management
* User registration and authentication
* Book rental and return processing
* Rental history tracking
* Administrative functions
* Real-time availability updates

**2. Architecture**

**System Architecture**

The system implements a microservices architecture with the following components:

* **Microservices**
* Book Service (FastAPI)
* User Service (FastAPI)
* Frontend (Streamlit)
* **Communication**
* REST APIs between services
* JWT-based authentication
* Asynchronous operations
* **Data Storage**
* SQLite databases (one per service)
* SQLAlchemy ORM for database operations

**3. Services**

**Book Service**

* **Port**: 8000
* **Responsibilities**:
* Book inventory management
* Availability tracking
* Book metadata management

**User Service**

* **Port**: 8001
* **Responsibilities**:
* User management
* Authentication
* Rental processing
* Rental history tracking

**Frontend Application**

* **Port**: 8501
* **Responsibilities**:
* User interface
* Service integration
* Real-time updates

**4. API Reference**

**Book Service API**

*Books Endpoints*



**User Service API**

*Authentication*



*Users*



*Rentals*



**5. Database Schema**

**Book Service Database**

**CREATE** **TABLE** books (  
 **id** STRING **PRIMARY** **KEY**,  
 title STRING,  
 author STRING,  
 genre STRING,  
 available\_copies INTEGER  
)

**User Service Database**

**CREATE** **TABLE** users (  
 **id** STRING **PRIMARY** **KEY**,  
 name STRING,  
 email STRING **UNIQUE**,  
 hashed\_password STRING,  
 rented\_books JSON,  
 is\_admin BOOLEAN  
)  
  
**CREATE** **TABLE** rentals (  
 **id** STRING **PRIMARY** **KEY**,  
 user\_id STRING **REFERENCES** users(**id**),  
 book\_id STRING,  
 book\_title STRING,  
 book\_author STRING,  
 rental\_date DATETIME,  
 return\_date DATETIME  
)

**6. Authentication & Authorization**

**Authentication Flow**

* User provides credentials
* User Service validates credentials
* JWT token generated with user details
* Token used for subsequent requests

**Security Features**

* Password hashing using bcrypt
* JWT-based authentication
* Role-based access control
* Token expiration
* Secure password storage

**7. Frontend Application**

**Features**

* Responsive UI
* Real-time updates
* Session management
* Role-based interface
* Error handling

**Pages**

* Login/Register
* Available Books
* My Rentals
* Rental History
* Admin Panel (for admins)

**8. Deployment Guide**

**Prerequisites**

* Docker
* Docker Compose
* Git

**Deployment Steps**

* Clone repository
* Configure environment variables
* Build containers:
* docker-compose build
* Start services:
* docker-compose up

**9. Docker Guide**

**9.1 Book Service Dockerfile**



**9.2 User Service Dockerfile**



**9.3 Frontend Docker File**



**Environment Variables**

BOOK\_SERVICE\_URL=http://book\_service:8000  
USER\_SERVICE\_URL=http://user\_service:8001  
SECRET\_KEY=your\_secret\_key\_here  
DATABASE\_URL=sqlite+aiosqlite:///./test.db

**10. Development Guide**

**Setup Development Environment**

* Create virtual environment
* Install dependencies:
* pip install -r requirements.txt
* Configure environment variables
* Start services individually



**Testing**

* Unit tests for each service
* Integration tests
* API tests using pytest
* Frontend tests

**11. Troubleshooting**

**Common Issues**

* Service connectivity problems
* Check network configuration
* Verify service URLs
* Check Docker network
* Database issues
* Verify database files exist
* Check permissions
* Validate schema migrations
* Authentication problems
* Check token expiration
* Validate user credentials

**Logging**

* Each service maintains its own logs
* Docker logs available via:
* docker-compose logs [service\_name]