# ព្រះរាជាណាចក្រកម្ពុជា
# ជាតិ សាសនា ព្រះមហាក្សត្រ

## Institute of Technology of Cambodia
## Department of Information of Technology

### Group: I4-GIC-C
### Course : Software Engineering

**Topic : Library Management System - Project Report Week 4**

| Name Student | ID of student | Score |
|---|---|---|
| RIN NAIRITH | e20221557 | ..... |
| THOU LAIHENG | e20220843 | ..... |

Lecturer: **Mr. ROUEN Pacharoth**

**Academic year : 2025-2026**

# 1. Project Overview

This report documents the significant progress made since Week 3, focusing on implementing a comprehensive Contact/Support Management System, Book Copy Management System, Audit System, Fine Management, features. The project has evolved into a fully functional library management system with proper authentication, authorization, and complete borrowing workflow. All features are documented through Swagger/OpenAPI and follow industry best practices.

# 2. Book Copy Management System (MAJOR FEATURE)

### 2.1 Enhanced Borrowing System

**Updated Borrow Controller:**

- New endpoint: POST `/api/borrows/book/{bookId}` - Borrow first available copy
- Automatically selects an available copy from the book's inventory
- Supports configurable borrow duration (default 14 days)
- Implements borrowing limit (max 5 books per user)

**Borrow Workflow:**

1. User selects a book to borrow
2. System finds first available copy
3. Marks copy as unavailable
4. Creates borrow record with due date
5. Sends notification to user
6. Logs action in audit trail

# 3. Book Copy Management System (MAJOR FEATURE)

### 3.1 Audit Log System

Complete audit trail for system actions:

- Logs all user actions (borrow, return, create, update, delete)
- Tracks admin operations
- Stores: user, action, entity type, timestamp
- Supports audit reporting and compliance

**Entity:**

- Extends BaseEntity for automatic timestamps
- Fields: userId, action, entityType, entityId, details

# 4. Fine Management System

### 4.1 Fine Service Implementation

Automated fine calculation and management:

**File:** `FineService.java`

- Calculates fines for overdue books
- Configurable daily fine rate
- Fine waiver capability (Admin)

**Entity:** `Fine.java`

- Fields: borrowRecord, amount, status, paidAt
- Links to specific borrow record
- Tracks payment status

**Controller:** `FineController.java`

- GET `/api/fines/my` - User's fines
- POST `/api/fines/{id}/pay` - Process payment
- PUT `/api/fines/{id}/waive` - Admin waive fine
- GET `/api/fines/all` - Admin view all fines

**Business Logic:**

- Auto-calculate fine on overdue return
- Daily rate: Configurable (e.g., $1/day)
- Grace period support
- Payment validation

## 5. User Role Management Enhancement

### 5.1 Librarian Promotion to Admin

**File Created:** `V6__promote_librarian_to_admin.sql`

Implemented database migration to grant admin privileges:

- Ensures ADMIN role exists in the roles table
- Grants ADMIN role to librarian user (librarian@lms.com)
- Avoids duplicate role assignments with NOT EXISTS checks
- Maintains existing LIBRARIAN role (multi-role support)
- Properly handles role-based authorization

**Business Logic:**

- Allows librarians to have both LIBRARIAN and ADMIN roles
- Enables access to admin-level contact management
- Maintains proper role hierarchy
- Supports Spring Security's role-based access control

## 6. Comprehensive API Documentation

### 6.1 Contact API Documentation

**File Created:** `CONTACT_API.md`

Complete API reference including:

- **Endpoint descriptions**: All user and admin endpoints
- **Request/response examples**: JSON format samples
- **Security requirements**: Authentication and authorization
- **Status values**: All contact status enums
- **Database schema**: Table structure documentation
- **Error handling**: HTTP status codes and error scenarios

Key sections:

- User Contact Endpoints (create, view own)
- Admin Contact Endpoints (view all, update, delete)
- Contact status values and workflow
- Contact response format specification
- Database schema details
- Error handling scenarios (404, 400, 401)

---

## 7. Enhanced Swagger/OpenAPI Integration

### 7.1 Implementation Details

- **Swagger UI:** Accessible at `http://localhost:8080/swagger-ui/index.html`
- **OpenAPI JSON:** Available at `http://localhost:8080/api-docs`
- **Dependency:** SpringDoc OpenAPI 2.7.0

### 7.2 Features Configured

- Custom API information (title, description, version)
- JWT Bearer token authentication support
- Server configuration
- Operations and tags sorting
- Package scanning for controllers
- Public access to Swagger endpoints

### 7.3 Enhanced Controller Documentation

All controllers now include:

- `@Tag` annotations for API grouping
- `@Operation` annotations with summaries and descriptions
- `@ApiResponses` for all possible HTTP responses
- `@Parameter` annotations for request parameters
- `@SecurityRequirement` for authentication needs
- Schema definitions for request/response models

---

## 8. Current System Components

**API Layers**

1. **User Management APIs** (`/api/users`)
    - CRUD operations
    - User status management
    - Role assignment
2. **Authentication APIs** (`/api/login`, `/api/register`)
    - JWT-based authentication
    - User registration
    - Login/logout

functionality

3. **Book Management APIs**
    - Book CRUD (`/api/books`)
    - Book Copy Management (`/api/book-copies`)
4. **Borrowing APIs** (`/api/borrows`)
    - Borrow books by copy or book ID
    - Return books
    - View borrowing history
    - Librarian management endpoints
5. **Reservation APIs** (`/api/reservations`)
    - Create book reservations
    - View reservation queue
    - Cancel reservations
6. **Fine Management APIs** (`/api/fines`)
    - Calculate overdue fines
    - Process payments
    - Admin fine waiver
7. **Notification APIs** (`/api/notifications`)
    - User notifications
    - Mark as read
    - Notification history
8. **Audit APIs** (`/api/audit`)
    - System audit logs
    - User action tracking
    - Compliance reporting

9. **Contact Management APIs**
    - User contact endpoints (`/api/contacts`)
    - Admin contact endpoints (`/api/admin/contacts`)
10. **System APIs** (`/api/system`)
    - System status
    - Health checks
    - Documentation links

#### Security Architecture

- **JWT Authentication**: Token-based authentication
- **Role-Based Access Control**: ADMIN, LIBRARIAN, STUDENT roles
- **Method-level Security**: `@PreAuthorize` annotations
- **Password Encryption**: BCrypt password encoder
- **Public Endpoints**: Swagger UI, login, register, static assets
- **Protected Endpoints**: All other APIs require JWT token

## 9. Project Structure Updates

### New Files Added (Since Week 3)

#### Database Migrations

```
src/main/resources/db/migration/
├── V5__create_contacts_table.sql
├── V6__promote_librarian_to_admin.sql
└── V7__add_book_copies.sql
```

#### Java Source Files - Book Copy System

```
src/main/java/com/demo/lms/
├── controller/book/
│   └── BookCopyController.java
├── service/book/
│   └── BookCopyService.java
├── repository/
│   └── BookCopyRepository.java
├── model/entity/
│   └── BookCopy.java
└── dto/
    ├── request/BookCopyRequest.java
    └── response/BookCopyResponse.java
```

#### Java Source Files - Contact System

```
src/main/java/com/demo/lms/
├── controller/contact/
│   ├── ContactController.java
│   └── AdminContactController.java
├── service/contact/
│   ├── ContactService.java
│   └── ContactServiceImpl.java
├── repository/
│   └── ContactRepository.java
├── model/entity/
│   └── Contact.java
└── dto/
    ├── request/
    │   ├── CreateContactRequest.java
    │   └── UpdateContactStatusRequest.java
    └── response/
        └── ContactResponse.java
```

**Frontend HTML Files**

```
src/main/resources/static/
├── addBookCopies.html
├── borrowBooks.html (Enhanced)
├── manageStudentBorrowings.html
├── myBorrowings.html
├── resetBorrowStats.html
└── js/
    ├── borrowBooks.js (Enhanced)
    ├── myBorrowings.js
    └── dashboard.js
```

**Documentation Files**

```
document/
├── CONTACT_API.md
├── AUTH_SETUP.md (existing)
├── SWAGGER_IMPLEMENTATION.md (existing)
├── USER_CRUD_API.md (existing)
├── ERROR_HANDLING.md (existing)
├── QUICK_START.md (existing)
└── README.md (existing)
```

---

# 10 .Technical Improvements

## 1. Error Handling

- Custom exception: `ContactNotFoundException`
- Integrated with `GlobalExceptionHandler`
- Consistent error response format
- Proper HTTP status codes

## 2. Validation

- Jakarta validation annotations (`@Valid`, `@NotNull`, `@Size`)
- Input validation in DTOs
- Business logic validation in services

## 3. Logging

- SLF4J with Lombok `@Slf4j`
- Comprehensive logging at INFO and DEBUG levels
- Error logging with stack traces
- Request/response logging in controllers

## 4. Transaction Management

- `@Transactional` for write operations
- `@Transactional(readOnly = true)` for read operations
- Proper isolation and propagation

## 5. Code Quality

- Consistent naming conventions
- Proper separation of concerns (Controller → Service → Repository)
- Builder pattern for response objects
- Comprehensive JavaDoc comments
- Clean code principles

---

# 11. Testing & Verification

## API Testing Capabilities

1. **Swagger UI**: Interactive API testing at http://localhost:8080/swagger-ui/index.html
2. **Manual Testing**: Postman/cURL compatible
3. **Authentication**: JWT token support in Swagger

## Test Scenarios Covered:

- Create and manage book copies
- Borrow books (by book ID or copy ID)
- Return borrowed books
- View borrowing history

- Calculate and manage fines
- Create and manage reservations
- Notification system
- Audit log tracking
- Create contact requests
- Admin contact management
- User authentication and authorization
- Role-based access control
- Borrowing limit enforcement (max 5 books)
- Availability tracking

## 12. Database Schema Status

### Tables Implemented

1. **users** - User accounts
2. **roles** - System roles (ADMIN, LIBRARIAN, STUDENT)
3. **user_roles** - Many-to-many relationship
4. **books** - Book catalog
5. **book_copies** - Physical book copies (inventory tracking) ☑ NEW
6. **borrow_records** - Borrowing transactions (renamed from borrowings)
7. **fines** - Overdue fine management ☑ NEW
8. **audit_logs** - System audit trail ☑ NEW
9. **contacts** - Support requests ☑ IMPLEMENTED

### Relationships

- users → contacts (one-to-many)
- users → borrow_records (one-to-many)
- users → reservations (one-to-many)
- users → fines (one-to-many via borrow_records)
- users → notifications (one-to-many)
- users → audit_logs (one-to-many)
- books → book_copies (one-to-many)
- book_copies → borrow_records (one-to-many)
- books → reservations (one-to-many)
- borrow_records → fines (one-to-one)
- users → roles (many-to-many via user_roles)

---

## 13. Build & Deployment Status

### Build Configuration

- **Build Tool**: Maven
- **Java Version**: 17
- **Spring Boot Version**: Latest
- **Database**: MySQL/H2 (configurable)
-

**Migration Tool**: Flyway

### Build Commands

```
# Clean and package
./mvnw.cmd clean package -DskipTests

# Run application
java -jar target/LMS-0.0.1-SNAPSHOT.jar --server.port=8080
```

### Application Status

- Build: SUCCESS
- Database Migrations: All 7 migrations applied successfully
- Server: Running on port 8080
- Swagger UI: Accessible and fully documented
- Authentication: JWT working
- All Major Features: Operational
- Book Copy System
- Borrowing & Returns
- Fine Management
- Audit Logging
- Contact Support

### Feature Completion

- Book Copy Management System
- Borrowing & Returns System
- Reservation System
- Fine Management
- Notification System
- Audit Logging
- Contact Management System
- User Role Enhancement
- Frontend Interfaces
- API Documentation

### New Capabilities Added

1. **Book Copy Management**: Track individual physical copies, not just book titles
2. **Complete Borrowing System**: Full workflow with availability tracking
3. **Automated Fine Calculation**: Overdue books generate automatic fines
4. **Audit Trail**: Complete system audit logging for compliance
5. **Admin Management Tools**: Professional interfaces for librarians
6. **Borrowing Limits**: Enforce maximum 5 books per user
7. **Multi-Role Support**: Users can have multiple roles simultaneously
8. **Enhanced Authorization**: Comprehensive role-based endpoint protection

# 15. Documentation Coverage

## Complete Documentation Set

1. **CONTACT_API.md**
    - Contact endpoint reference
    - Request/response formats
    - Status workflow
    - Error scenarios
2. **AUTH_SETUP.md**
    - Authentication implementation
    - JWT configuration
    - Security matrix
3. **SWAGGER_IMPLEMENTATION.md**
    - Swagger setup guide
    - API testing instructions
    - Authentication in Swagger
4. **USER_CRUD_API.md**
    - User management endpoints
    - CRUD operations
    - Status management
5. **ERROR_HANDLING.md**
    - Error response structure
    - Exception types
    - HTTP status codes
6. **QUICK_START.md**
    - 5-minute setup guide
    - Prerequisites
    - First API test
7. **README.md**
    - Project overview
    - Getting started
    - Documentation index

# 16. Key Achievements

## 1. Production-Ready Library Management System

- Complete book copy tracking and inventory management
- Full borrowing lifecycle with automated workflows
- Reservation system with queue management
- Automated fine calculation and payment processing
- Real-time availability tracking

## 2. Supporting Systems

- Comprehensive notification system for user communication
- Complete audit trail for compliance and reporting
- Support ticket system for user assistance
- Admin tools for efficient library management

## 3. Enhanced User Management

- Multi-role support with flexible permissions
- Role-based access control at method level
- Proper authorization hierarchy
- Secure authentication with JWT

## 4. Professional API Design

- RESTful conventions throughout
- Consistent response formats
- Proper HTTP status codes
- Complete Swagger/OpenAPI documentation
- 25+ well-documented endpoints

## 5. Code Quality & Architecture

- Clean architecture with clear separation of concerns
- SOLID principles implementation
- Comprehensive logging and error handling
- Transaction management for data consistency
- Professional exception handling strategies

# 17. Future Enhancements (Recommendations)

## Short-Term

1. **Email Integration**: Send email notifications for important events
2. **Book Reviews**: Allow users to rate and review books
3. **Advanced Search**: Full-text search with filters
4. **Export Reports**: PDF/Excel reports for librarians
5. **Book Categories**: Enhanced category management

### Medium-Term

1. **Analytics Dashboard**: Visual statistics and insights
2. **Barcode Scanning**: Mobile barcode integration
3. **Late Fee Automation**: Auto-calculate and send reminders
4. **Waiting List**: Auto-notify when reserved books available
5. **User Preferences**: Customizable notification settings

### Long-Term

1. **Mobile Applications**: iOS and Android native apps
2. **AI Recommendations**: Book recommendation engine
3. **Integration with Library Systems**: Connect to external catalogs
4. **Digital Content**: eBook and audiobook support
5. **Community Features**: Reading groups and forums

## 18. Learning Outcomes

### Technical Skills Demonstrated

1. **Spring Boot Framework**: Controllers, services, repositories pattern
2. **Spring Security**: JWT authentication and role-based authorization
3. **JPA/Hibernate**: Complex entity relationships and custom queries
4. **Flyway Migrations**: Database version control and evolution
5. **Swagger/OpenAPI**: Comprehensive API documentation
6. **RESTful API Design**: Industry best practices and conventions
7. **Error Handling**: Global exception handling strategies
8. **Transaction Management**: ACID compliance and data consistency
9. **Logging & Monitoring**: SLF4J implementation
10. **Frontend Development**: HTML, CSS, JavaScript integration
11. **Git Version Control**: Code management

### Software Engineering Practices

- Clean code principles
- SOLID principles
- Design patterns (Builder, Repository, Service)
- API-first development
- Documentation-driven development
- Test-driven development mindset

### System Capabilities Summary

**User Capabilities**

- Register and login
- Submit contact/support requests
- View own contact history
- Borrow books

- View book catalog

**Librarian Capabilities**

- All user capabilities
- View all contact requests
- Update contact status
- Respond to contacts
- Delete contacts
- Manage book inventory

**Admin Capabilities**

- All librarian capabilities
- User management (CRUD)
- Role assignment
- System configuration
- Full contact management

---

# 19. Conclusion

Week 4 has been highly productive with the successful implementation of a complete Contact/Support Management System. This feature significantly enhances the Library Management System by providing a structured way for users to communicate with administrators and request support. The implementation follows industry best practices with proper authentication, authorization, error handling, and comprehensive documentation.

The multi-role support enhancement allows for more flexible user management, enabling staff members to have multiple roles simultaneously. This architectural improvement provides better scalability and role management for future features.

All features are production-ready with full Swagger documentation, proper error handling, and comprehensive logging. The codebase maintains high quality standards with clean architecture, proper separation of concerns, and adherence to SOLID principles.

## Borrowing Lifecycle

1. User borrows book → Status: BORROWED
2. System updates copy availability
3. Notification sent to user with due date
4. User returns book → Status: RETURNED
5. If overdue, fine is generated automatically

# 20. References

**Technical Documentation & Frameworks**

1. **Spring Framework Documentation**
   Spring Team. (2024). *Spring Boot Reference Documentation (Version 3.x)* Pivotal Software, Inc.
   Available at: https://docs.spring.io/spring-boot/docs/current/reference/html/
   *Used for: Application framework, dependency injection, REST API development*

2. **Spring Security Reference**
   Spring Security Team. (2024). *Spring Security Reference (Version 6.x)*. Pivotal Software, Inc.
   Available at: https://docs.spring.io/spring-security/reference/index.html
   *Used for: JWT authentication, role-based authorization, method-level security*

3. **Spring Data JPA Documentation**
   Spring Data Team. (2024). *Spring Data JPA - Reference Documentation*. Pivotal Software, Inc.
   Available at: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/
   *Used for: Database operations, repository pattern, entity relationships*

4. **Flyway Database Migration**
   Redgate. (2024). *Flyway Documentation - Database Migrations Made Easy*.
   Available at: https://flywaydb.org/documentation/
   *Used for: Database version control, schema migrations*

5. **SpringDoc OpenAPI (Swagger)**
   SpringDoc Team. (2024). *SpringDoc OpenAPI Documentation*.
   Available at: https://springdoc.org/
   *Used for: API documentation, Swagger UI integration, OpenAPI 3.0 specification*

**Design Patterns & Architecture**

6. **Martin, R. C.**(2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*.
   Prentice Hall. ISBN: 978-0134494166
   *Used for: Separation of concerns, dependency rule, clean code principles*

7. **Gamma, E., Helm, R., Johnson, R., & Vlissides, J.** (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*.
   Addison-Wesley Professional. ISBN: 978-0201633610
   *Used for: Repository pattern, Builder pattern, Service layer pattern*

**RESTful API Design**

8. **Fielding, R. T.** (2000). *Architectural Styles and the Design of Network-based Software Architectures*.
   Doctoral dissertation, University of California, Irvine.
   Available at: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
   **Used for: REST principles, HTTP methods, resource-oriented design*

9. **Richardson, L., & Ruby, S.**(2013). *RESTful Web APIs*.
   O'Reilly Media. ISBN: 978-1449358068
   *Used for: API design best practices, HTTP status codes, endpoint structure*

**Database Design & SQL**

10. **Hernandez, M. J.**(2013). *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design* (3rd ed.).
    Addison-Wesley Professional. ISBN: 978-0321884497
    *Used for: Database normalization, entity relationships, schema design*

**Authentication & Security**

11. **Jones, M., Bradley, J., & Sakimura, N.** (2015). *JSON Web Token (JWT) - RFC 7519*
    Internet Engineering Task Force (IETF).
    Available at: https://tools.ietf.org/html/rfc7519
    *Used for: JWT token structure, claims, authentication implementation*

12. **OWASP Foundation.** (2024). *OWASP Top Ten Web Application Security Risks*
    Available at: https://owasp.org/www-project-top-ten/
    *Used for: Security best practices, vulnerability prevention*

**Frontend Development**

13. **Mozilla Developer Network (MDN).**(2024). *HTML, CSS, and JavaScript Documentation*.
    Available at: https://developer.mozilla.org/
    *Used for: Frontend development, DOM manipulation, AJAX requests*

**Tools & Technologies**

14. **Apache Maven Documentation.**
    Apache Software Foundation. (2024). *Maven - Welcome to Apache Maven*.
    Available at: https://maven.apache.org/guides/index.html
    *Used for: Project build management, dependency management*

15. **H2 Database Engine.**
    H2 Database Team. (2024). **H2 Database Documentation**.
    Available at: https://www.h2database.com/html/main.html
    *Used for: Development database, testing, SQL compliance*