1. Project Overview:

- **Introduction**: Briefly explains the purpose of the system and its main functionalities.
- **Features**: Lists the features offered by the system, including user authentication, book management, borrowing, user management, reporting, and analytics.

2. Setup Instructions:

- Provides step-by-step instructions for setting up the project environment, including cloning the repository, importing into Eclipse IDE, configuring Tomcat Server, setting up MySQL Database, and modifying configuration files.

3. User Guide:

- Describes how to use the system, including login instructions for admins and students, managing books, managing users (admins only), and borrowing books.

4. Developer Guide:

- Offers insight into the project structure, key components, code overview, configuration details, and the database schema.
- **Project Structure**: Explains the organization of source files, resources, and databases.
- **Key Components**: Describes the main components such as controllers, services, repositories, and models.
- **Code Overview**: Highlights important Java classes responsible for handling requests, business logic, and database operations.
- **Configuration**: Specifies configuration files and their purposes.
- **Database Schema**: Provides an overview of the database structure.

5. API Documentation:

- Documents the endpoints of the API, including operations for books, users, and borrowing activities.
- Provides request/response examples for better understanding.

6. Maintenance and Support:

- Lists common issues and their solutions, along with contact information for further assistance.

eLibrary Management System

Project Documentation

1. Project Overview

Introduction

The eLibrary Management System is designed to help manage library resources efficiently. It provides functionalities for adding, updating, deleting, and searching for books. It also manages user accounts and borrowing activities.

Features

```sql
- User authentication (student and admin)
- Book management (add, update, delete, search)
- Borrow and return books
- User management (add, update, delete users)
- Reporting and analytics
```

System Requirements

```
- Operating System: Windows/Linux/MacOS
- Java JDK 8 or higher
- Apache Tomcat 8.5 or higher
- MySQL 5.7 or higher
```

```
- Eclipse IDE or any other Java IDE
```

Setup Instructions

1. Clone the repository: `git clone <repository_url>`

2. Import the project into Eclipse IDE:

- Open Eclipse IDE

- Go to File > Import > Existing Projects into Workspace

- Select the cloned repository

3. Configure Tomcat Server:

- Add Tomcat server in Eclipse

- Add the eLibrary Management project to the server

4. Setup MySQL Database:

- Create a new database `library_db`

- Execute the SQL script provided in the `database` folder to create necessary tables

5. Modify configuration files:

- Update database configurations in `src/main/resources/application.properties`

6. Run the project:

- Start the Tomcat server from Eclipse

- Access the application at `http://localhost:8080/ELibraryManagementSystem`

3. User Guide

Using the System

1. Login as Admin or Student
   - Admin can manage books and users
   - Students can search for books and manage their borrowing activities

2. Managing Books
   - Add new books
   - Update book details
   - Delete books
   - Search for books

3. Managing Users (Admin only)
   - Add new users

- Update user information
- Delete users

4. Borrowing Books
- Search for available books
- Borrow books
- Return books

4. Developer Guide

Project Structure

- `src/main/java`: Contains Java source files
- `src/main/resources`: Contains configuration files
- `webapp`: Contains JSP files and static resources
- `database`: Contains SQL scripts for database setup

Key Components

- Controllers: Handle HTTP requests
- Services: Business logic
- Repositories: Database interactions
- Models: Entity classes representing database tables

Code Overview

- `BookController.java`: Handles book-related requests
- `UserController.java`: Handles user-related requests
- `BookService.java`: Business logic for books
- `UserService.java`: Business logic for users
- `BookRepository.java`: Database operations for books
- `UserRepository.java`: Database operations for users

Configuration

- `application.properties`: Database and other configurations
- `web.xml`: Servlet configurations

Database Schema

- `library_db`
  - `books` table: Stores book details
  - `users` table: Stores user details
  - `borrow_records` table: Stores borrowing information

5. API Documentation

Endpoints

- `/books`
  - GET: Retrieve all books
  - POST: Add a new book
  - PUT: Update book details
  - DELETE: Delete a book

- `/users`
  - GET: Retrieve all users
  - POST: Add a new user
  - PUT: Update user details
  - DELETE: Delete a user

- `/borrow`
  - POST: Borrow a book
  - PUT: Return a book

Request/Response Examples

- Retrieve all books:

GET /books Response: 200 OK

[{ "id": 1, "title": "Book Title", "author": "Author Name", ... }, ...]

- Add a new book:

POST /books Request:

{ "title": "New Book", "author": "Author Name", ... } Response: 201 Created

6. Maintenance and Support

Common Issues and Solutions

- Database connection issues:
  - Ensure the database server is running
  - Verify database configurations in `application.properties`
- Server startup issues:
  - Check Tomcat logs for errors
  - Ensure all dependencies are properly configured
-
- Contact Info
- For further assistance, please contact Swapnil raibole at swapnilraibole107@gmail.com