Shelf Masters

Library Management System
Version 1.0

Revision History

Date	Version	Description	Author
1/Mar/25	1.0	Initial Database Requirements Specifications document	Abhiroop Goel
2/Mar/25	1.1	Technical requirement updates and DB Diagram	Spencer Sliffex

Table of Contents

Database Requirements Specifications

1. Introduction

The Database Requirements Specification serves as a structured blueprint delineating the anticipated functionalities, requirements, and expectations of the database application. This document offers a meticulous overview of the program's scope and intricacies, facilitating a comprehensive understanding prior to the commencement of development activities.

1.1 Purpose

The purpose of the DRS is to comprehensively describe what the software is expected to do. It will also list the components required for software functionality. Additionally, the SRS will cover essential factors such as nonfunctional requirements and design constraints that are crucial for software development

1.2 Scope

The database will cover book and media cataloging, membership management, borrowing and returning transactions, reservation handling, and fine calculations. It will not include external vendor integration or e-book lending functionalities.

1.3 Definitions, Acronyms, and Abbreviations

- LMS: Library Management System

- ISBN: International Standard Book Number

- BCNF: Boyce-Codd Normal Form

- DDL: Data Definition Language

- NTP: Network Time Protocol

1.4 References

Title	Date	Source	Link
EECS447: Project Description	Jan 17, 2025	EECS 447 Lecture on Canvas	EECS 447 Project Description

1.5 Overview

Overall Description

Describes the general factors and requirements for the database as well as background for the requirements

Specific Requirements

Describes all database requirements including functional and supplementary requirements, and use-cases.

Classification of Functional Requirements	List of the functional requirements indicating their type.

1.6   Stakeholders

- Library Staff: Manage book checkouts, returns, fines, and member accounts.
- Library Members: Borrow books, check due dates, and pay fines.
- Administrators: Oversee database performance, security, and system updates.
- Developers: Maintain and improve the system's functionality.

2.   Requirements

2.1   Functional Requirements

- User Administration: Add, update, delete members and staff accounts.
- Book and Media Management: Maintain a catalog of books, magazines, and digital media.
- Borrowing & Returning Transactions: Track loaned items and return dates.
- Reservation System: Allow members to reserve checked-out items.
- Fine Calculation: Apply fees based on overdue items and membership type.
- Notifications: Alert members about due dates, overdue items, and reserved item availability.
- Report Generation:
    - Books due soon
    - Members with overdue books
    - Frequent borrowers by genre
    - Library revenue from fines
    - Monthly lending statistics
    - Underutilized books

2.2   Data Entities

- Books (ISBN, Title, Author, Year, Genre, Availability)
- Magazines (Title, Issue Number, Publication Date, Availability)
- Digital Media (Title, Creator, Format, Availability)
- Members (ID, Name, Contact Info, Membership Type, Account Status)
- Borrowing Transactions (Transaction ID, Member ID, Item ID, Borrow Date, Due Date, Return Date, Fine)
- Reservations (Reservation ID, Member ID, Item ID, Request Date, Status)

2.3   Non-Functional Requirements

- Performance: Query execution should not exceed 2 seconds for common operations.
- Security: Role-based access control for members and staff.
- Compliance: GDPR compliance for user data protection.

2.4 Hardware and Software Requirements

- Database Software: Data Grip, SQLite, PostgreSQL, DBeaver, Azure DB/AWS DB(TBD)
- Server Requirements: Were using a Postgres server that is running on linux. We will be deploying our db updates using github workflows. It'll be in the east region I believe, maybe central Canada… don't really care since we are not deploying this app. Running with username and password logins since this isn't for production.
- Client Interface: Multi tenant: has client faced UI and admin faced UI as well as super admin UI.
- Development Tools: Django, SQLite, Postgre, Azure/AWS, Github, Pycharm, Datagraip, and a JS or TS front end framework

3. Appendices

https://github.com/spencer-sliffe/EECS-447-Project