TABLE OF CONTENTS

1. INTRODUCTION

1.1Purpose

This is the Software Requirements Specification (SRS) for the Library Management System. The purpose of this document is to convey information about the application's requirements, both functional and non-functional, to the reader. This document provides (a) a description of the environment in which the application is expected to operate, (b) a definition of the application's capabilities, and (c) a specification of the application's functional and nonfunctional requirements.

The document is intended to serve several groups of audiences:

- First, it is anticipated that the SRS will be used by the application designers. Designers will use the information recorded here as the basis for creating the application's design.
- Second, the client for the project, the library manager in our case, is expected to review this document. The SRS will serve to establish a basis for agreement between the client and development team about the functionality to be provided by the application.
- Third, the application maintainers will review the document to clarity their understanding of what the application does.
- Fourth, test planners will use this document to derive test plans and test cases. Finally, the project manager will use this document during project planning and monitoring

1.2 Scope

The Library Management System is a comprehensive platform designed to handle the full spectrum of library tasks. The system allows for the cataloging of books, tracking borrowing/returning transactions, maintaining user records, generating automated alerts for due books, and producing reports for management. It supports role-based access, meaning that different types of users (like students, librarians, and administrators) will have different permissions and capabilities. The LMS will be developed as a web-based system, ensuring accessibility from a variety of devices including desktops, tablets, and mobile phones. It is expected to enhance the efficiency, accuracy, and reliability of library operations.

The application will provide the following capabilities:

- The application will be access via a LAN on a PC terminal in the library
- Library staff will be able to manage library user accounts including remove, change, and add.
- Library staff will be able to manage the book inventory database including remove, change, and add.
- The application will record all books that are checked out, checked in, and recalled.
- The application will generate reports for administrative purposes.
- The application will provide search functions on books based on ISBN, subject, title, or author.

The project's client has determined that this application will provide the following benefits:

- Provide additional flexibility and convenience to the library users.
- Provide better reliability and security of the library information.
- Provide a more productive environment for the library staff member.
- Reduce the cost of the library operations.

1.3 Definitions, Acronyms, and Abbreviations

- **LMS:** Library Management System
- **UI:** User Interface
- **DBMS:** Database Management System
- **ISBN:** International Standard Book Number – a unique identifier for books
- **Admin:** Administrator – a user with the highest privileges in the system, typically responsible for managing users and settings
- **CRUD:** Create, Read, Update, Delete – standard database operations
- **OTP: One**-Time Password
- **SSL:** Secure Socket Layer
- **REST API:** Representational State Transfer – an architectural style for designing networked applications

- **SRS:** Software Requirements Specification
- **LAN:** Local Area Network

1.4 References

- IEEE 830-1998: Recommended Practice for Software Requirements Specifications
- MySQL Documentation for Database Design
- Django Documentation for Backend Implementation
- Bootstrap Documentation for Front-End Design
- Institutional IT Policy Documents

1.5 Overview

This document begins by describing the product in general terms, including its functionality and user types. It then delves into more specific system requirements, including both functional and non-functional requirements. The document includes design constraints, user interface details, performance expectations, and security needs. Each section is written to give developers clear guidelines and to ensure all stakeholders understand how the final system will function and behave. Section 2 offers a general description of the system, its users, and the environment. Section 3 breaks down the precise functional, interface, and quality attributes the system must fulfill.

2 OVERALL DESCRIPTION

2.1 Product Perspective

The Library Management System is a standalone, modular web application designed for educational institutions, public libraries, or corporate organizations. Though it functions independently, it can integrate with existing systems like student management systems or human resource databases through API interfaces. The system will have a layered architecture, separating the presentation layer (UI), business logic layer, and data access layer. This modularity enhances maintainability and supports future expansion.

2.2 Product Functions

The LMS provides a variety of functions, which include but are not limited to:

- **Book Cataloging**: Add new books, categorize them, and assign ISBNs.
- **Book Search**: Advanced search functionality by title, author, genre, or ISBN.
- **Member Registration**: Allows users to create and manage personal accounts.
- **Book Borrowing and Returning**: Tracks which books are borrowed, due dates, and return confirmations.
- **Notification System**: Sends reminders for due books or overdue fines.
- **Report Generation**: Create reports on book usage, member activity, late returns, and inventory.

2.3 User Classes & Characteristics

The system is intended to be used by multiple types of users, each with specific characteristics and needs:

Primary Users

These include students, faculty members, and regular library patrons who will primarily use the system to search for books, check availability, place holds, and view their borrowing history. They need a user-friendly interface with minimal learning curve.

a. Library Members:

- Characteristics: Students, faculty, or public members.
- Permissions: Search, view books, reserve, borrow, renew books, and pay fines.

Administrative Users

These are library staff responsible for the day-to-day management of library resources. They need more advanced tools for managing books, processing borrow/return requests, calculating fines, and managing user accounts.

a. Librarians and Library Staff:

- Characteristics: Trained personnel with authority to manage catalog and transactions.
- Permissions: Add/edit/delete books, manage members, handle issues/return, generate reports.

Secondary Users

These include IT staff who are responsible for maintaining the server infrastructure, backups, updates, and security. They may not interact directly with the system's UI but will manage deployment, configuration, and system integrity.

a. IT Administrators:

- Characteristics: Technical staff responsible for software and database maintenance.
- Permissions: System configuration, backups, security patches, API integrations.

2.4 Operating Environment

This section outlines the hardware, software, and network environments required for the Library Management System (LMS) to function effectively. The operating environment must support the development, deployment, and execution of the application, as well as meet performance, security, and scalability expectations.

1. Client-Side (User Access Environment)

End users—such as students, faculty, librarians, and administrators—will access the LMS through web browsers. The client environmental requirements include:

a. Supported Devices:

- Desktop Computers (Windows, macOS, Linux)
- Laptops
- Tablets (Android, iPad)
- Smartphones (iOS, Android)

b. Supported Web Browsers:

- Google Chrome (v90+)
- Mozilla Firefox (v85+)
- Microsoft Edge (v90+)
- Safari (v13+)

The LMS interface will be built using responsive web design techniques (e.g., Bootstrap) to ensure usability on both desktop and mobile devices.

c. Client-Side Technologies:

- HTML5
- CSS3
- JavaScript (ES6 or later)
- Frontend frameworks (e.g., Bootstrap, optional use of React or Vue)

Additional Requirements:

- Cookies and JavaScript must be enabled in the browser.
- Minimum screen resolution: 1024x768 pixels

2. Server-Side (Backend Environment)

The server-side environment is responsible for handling all core operations such as authentication, database access, book management, and reporting.

a. Web Server:

- Apache HTTP Server 2.4+
- Nginx (optional alternative)

b. Application Server / Runtime:

- For PHP-based LMS:
  - PHP 7.4 or above (recommended: PHP 8.x)
- For Python-based LMS:
  - Python 3.8 or above with Flask/Django framework

c. Operating System:

- Ubuntu Server 20.04 LTS or later (recommended)
- CentOS 8 / RHEL 8
- Windows Server 2019 or later (optional)

Linux-based servers are preferred for security, stability, and open-source tooling support.

3. Database Management System (DBMS)

The LMS will use a relational database to manage structured data such as book records, user accounts, and transaction logs.

a. Supported DBMS:

- MySQL 8.0 (preferred)
- PostgreSQL 13 or later
- SQLite (for small-scale/local installations)

b. Database Features Required:

- ACID compliance
- Support for foreign keys and transactions
- Regular backup and restore capabilities
- User-based access control

4. Network Environment

The system should operate in a network environment that ensures reliable communication between clients and the server.

a. Deployment Options:

- Local area network (LAN) deployment within an institution
- Cloud hosting (AWS, Azure, Google Cloud)
- Shared hosting or VPS

b. Protocols and Standards:

- HTTP/HTTPS (TLS 1.2 or higher for secure communication)
- SMTP for sending email notifications

c. Bandwidth and Latency:

- Minimum recommended internet speed for client devices: 2 Mbps
- Server should have at least a 10 Mbps dedicated line with low latency

2.5. Design and Implementation Constraints

The current hardware for the existing Library Book System is mainframe with text type terminal. Therefore, if the new system is PC based, there will be a need to replace it with PC hardware and new network facility. The Library Book System can potentially have hundreds of users. It is unrealistic to provide training for everyone. Therefore, the system should be designed for easy to use, providing help instructions, and appropriate error messages for invalid user inputs. Security is important for library operation. Library users are allowed to use the Library Book System only for searching book records. Users should never be able to break into the system and to perform any modification. Reliability is vital to library operation. The Library Book System should not have any unscheduled down time during library operation hours. Any down time in operation hours has significant impact to the operation and cause inconvenience to everyone in library. It should contain the following points:

- Must adhere to open-source licensing policies
- Should be responsive and mobile-friendly
- Must be scalable and modular

- Integration should be possible with APIs for future expansion (e.g., payment gateway, university portals)

2.6 User Documentation

Comprehensive Documentation Suite

The following documents will be provided:

- **User Manual**: A step-by-step guide for library patrons
- **Administrator Guide**: Detailed documentation for library staff
- **Developer Guide**: Instructions for future developers, including system architecture
- **Installation Guide**: Setup and deployment instructions for IT personnel

2.7 Assumptions & Dependencies

This section provides detailed assumptions that have been made during the design and development of the Library Management System (LMS), as well as the external dependencies that the system requires for proper operation. These must be considered to ensure realistic expectations and smooth implementation.

➢ Assumptions:

1. **Users Have Access to Internet and Compactable Devices**
   It is assumed that all end users, including administrators, librarians, and members will access the system using a device (desktop, laptop, tablet, or smartphone) with a stable internet connection. Since LMS is web-based, lack of internet access would limit or prevent usage of the system.

2. **Users Possess Basic Computer and Web Literacy**
   The system assumes that users are capable of navigating a standard web interface. This includes the ability to:
   - Open and use a web browser

- Fill out and submit forms (e.g., login, book request)
- Understand common UI elements such as buttons, menus, and tables

3. **Book Lending and Return Rules Are Predefined and Uniform**

   It is assumed that lending policies (e.g., maximum number of issued books, lending period, fine rates for overdue returns) are consistent across the entire library system. The application logic is designed based on fixed parameters unless otherwise configured by the administrator.

4. **Librarian and Admin Roles Are Pre-assigned**

   It is assumed that administrative roles such as librarian and system administrator are designated before the system is used. Role-based access control will be enforced, and the system depends on having these user roles properly assigned.

5. **No Third-party Library Network Integration**

   The LMS is assumed to function independently without real-time integration with third-party or national library catalogs (e.g., WorldCat, Koha, or other consortiums). Inter-library loan systems and book-sharing networks are outside the current scope of this LMS.

6. **SMTP or Email Server Availability**

   The system assumes that a working email infrastructure (SMTP server or email API service) is configured and available to send:

   - Book due date reminders
   - Overdue notifications
   - Account alerts and confirmations

7. **User Data Is Valid and Verified**

   The system assumes that users provide valid and truthful information during registration (e.g., name, email, university ID). Verification processes may be implemented separately by the institution.

8. **System Will Be Managed by a Dedicated Administrator**

It is assumed that a system administrator will be assigned to manage:

- User account issues
- Librarian account creation
- Database and server maintenance
- General system supervision

➢ Dependencies

1. **Relational Database Management System (DBMS)**

   The LMS relies on a relational database (e.g., MySQL, PostgreSQL) to store all data including user records, book information, transactions, and fines. This dependency includes:

   - Proper installation and configuration of the DBMS
   - Continuous availability and backup routines
   - Secure access controls and data integrity constraints

2. **Web Server Software**

   A fully functional web server (such as Apache or Nginx) must be available to host the LMS. This server handles HTTP requests, serves web pages, and processes backend logic. The system's usability depends on:

   - Uptime and responsiveness of the web server
   - Correct server configurations (e.g., PHP settings, URL routing)
   - Security modules (e.g., SSL certificates for HTTPS)

3. **Email Sending Service (SMTP or API)**

   To enable notification functionalities (e.g., automated emails for due dates or password resets), the LMS depends on a configured and reliable email delivery system. Options include:

   - SMTP configuration (e.g., Gmail, Microsoft Outlook)
   - Email API services (e.g., SendGrid, Mailgun)

   If the email server is down or incorrectly configured, users may not receive important system alerts.

4. **Authentication Framework or Library**

The LMS depends on secure authentication mechanisms to validate user identities. This may include:

- Use of hashing algorithms (e.g., bcrypt, SHA-256) for password storage
- Session or token-based authentication for login management
- CAPTCHA integration for bot prevention (optional)

5. **Modern Web Browser Compatibility**

   The frontend of the LMS is built using modern HTML5, CSS3, and JavaScript technologies. The system requires that users access it through updated browsers that support:

   - JavaScript execution
   - Responsive design layouts
   - Cookies and local storage (if required for session management)

6. **Operating System and Hosting Environment**

   The system requires a hosting environment that supports:

   - A web server stack (e.g., LAMP: Linux, Apache, MySQL, PHP)
   - Operating systems like Ubuntu, CentOS, or Windows Server
   - Scheduled backup and cron jobs for automation (e.g., reminders, cleanup tasks)

7. **Regular System Maintenance and Updates**

   The stability and security of the LMS depend on:

   - Regular application updates
   - Database optimization and cleanup routines
   - Monitoring server logs and system activity

8. **Institutional Support**

   It is assumed that the hosting institution (e.g., school, college, university) will provide:

   - Hosting infrastructure (cloud or on-premises)
   - IT support staff for technical issues
   - Policy guidelines for book lending and user registratio.

3 SPECIFIC REQUIREMENTS

3.1 Functional Requirements

3.1.1 Book Management Module

- Add new books with attributes like title, author, ISBN, category, and location
- Update or delete book records
- View current availability status
- Support bulk upload of books via Excel or CSV files

3.1.2 Member Management Module

- Register new users with unique ID and contact info
- View and edit member profiles
- Assign different roles (student, faculty, librarian)
- Set borrowing limits and loan periods based on role

3.1.3 Transaction Module

- Process borrow and return transactions
- Update book status (borrowed, available, overdue)
- Automatically calculate due dates and applicable fines
- Allow renewal and reservation of books

3.1.4 Notification Module

- Send due date reminders
- Alert members about overdue books or reserved book availability
- Support email and SMS formats
- Allow users to opt-in or out of notifications

3.1.5 Reporting Module

- Generate daily, weekly, and monthly reports
- Track most borrowed books, overdue records, and active users
- Export reports in PDF and Excel formats

- Visualize data with charts and tables

3.2 External Interface Requirements

3.2.1 User Interfaces

- Clean and intuitive UI for different user roles
- Accessible design with support for screen readers
- Responsive layout for mobile and tablet users

3.2.2 Hardware Interfaces

- Integration with barcode scanners for book check-in/check-out
- Printers for generating transaction receipts

3.2.3 Software Interfaces

- Interface with relational database (MySQL/PostgreSQL)
- Integration with SMTP for emails and third-party SMS services

3.2.4 Communication Interfaces

- RESTful APIs for integration with external systems
- Secure data transfer via HTTPS
- Email communication via SMTP protocol

3.3 Performance Requirements

- Should support at least 500 concurrent users
- Page load time under 2 seconds
- High availability with minimum downtime
- System backups every 24 hours

3.4 Security Requirements

- **Authentication:** Multi-factor authentication for administrative accounts
- **Data Encryption:** All sensitive data encrypted using AES-256 standards
- **Access Control:** Role-based permissions prevent unauthorized system access
- **Audit Trail:** Complete logging of all user actions and system changes

3.5 Usability Requirements

- **Learning Curve:** New staff complete basic training within 4 hours
- **Error Prevention:** Input validation prevents common data entry mistakes
- **Accessibility:** WCAG 2.1 AA compliance for users with disabilities
- **Mobile Support:** Responsive design functions on tablets and smartphone

3.6 Logical Database Requirements

- Structured schema with normalization
- Entity relationships: Users, Books, Transactions, Fines, Reports
- Referential integrity and foreign key constraints
- Support for data archiving and rollback

3.7 Design Constraints

- Must use open-source technologies
- Follow MVC architecture pattern
- Use responsive web design principles
- Ensure accessibility (WCAG compliance)

3.8 Software System Attributes

3.8.1 Reliability

- Fail-safe mechanisms for transaction rollback
- Logging of all actions for audit purposes
- The system shall be recovered within 10 minutes if it is down.
- The system shall be recovered without intervention at user terminal if it is down.
- The system shall show appropriate messages at terminal when system is down.
- The system should have 99% reliability during library operating hours.

- Scheduled down time after library operating hours shall not be more than 1 hour per day.
- The system shall generate error messages when the user attempts to enter invalid data.

3.8.2 Availability

- 99.5% uptime
- Downtime limited to scheduled maintenance windows

3.8.3 Security

- Role-based access control
- Encrypted storage of sensitive data (e.g., passwords)
- Protection against SQL injections, XSS, and CSRF
- The account management system shall only be used by managers or users with defined privileges.
- The check-in, check-out and recall system shall only be used by users who have librarian ID.
- The Patron information report shall be generated by users who have librarian ID.
- The book signs out report or book purchase report shall only be generated by managers or users with defined privileges.
- Database update data shall be committed to the database only after the managers have approved

3.8.4 Maintainability

- Modular codebase for easy debugging and updates
- Comprehensive documentation for developers
- Use of version control (Git)

3.8.5 Portability

- Compatible with major operating systems
- Easily deployable on cloud platforms like AWS or Azure

3.8.6 Scalability

- Horizontal scaling supported (e.g., load balancing)
- Database sharding and caching is possible for large datasets
- Prepared for future enhancements (e.g., digital book integration)

4. Other Requirements

This section outlines additional technical, operational, and development-related requirements that are not covered in the functional or non-functional sections but are essential for the long-term sustainability, maintainability, and reliability of the Library Management System (LMS).

4.1 Regular Data Backups

To ensure data integrity, security, and recoverability, the system must support an automated and reliable data backup mechanism.

Details:

- **Backup Frequency:**
  Backups should be scheduled daily or at defined intervals (e.g., nightly at 2 AM) based on system usage. Critical data includes:
  - User records
  - Book inventory
  - Issuance and return logs
  - Transaction history
  - Fine and penalty records
- **Backup Storage:**
  Backups should be stored securely in a location separate from the production database. Options include:
  - Cloud storage (e.g., AWS S3, Google Cloud Storage)
  - External on-site servers or drives
  - Secure file transfer to remote backup servers
- **Backup Format:**
  Data should be exported in SQL dump format (e.g., .sql or .sql.gz) or a compressed JSON/XML format, depending on the database type.

- **Backup Rotation & Retention Policy:**
  Maintain a retention policy (e.g., retain daily backups for 7 days, weekly for 1 month) and rotate older backups to save storage.
- **Recovery Procedures:**
  The system must provide documented steps to restore the database from a backup, ensuring minimal downtime in the event of data loss or corruption.
- **Security:**
  Backup files must be encrypted and access-controlled to prevent unauthorized access or tampering.

4.2 Source Code Documentation and Version Control

The development and maintenance of the LMS must adhere to professional coding standards, including detailed documentation and the use of version control systems.

Details:

- **Source Code Documentation:**
  - All code should include meaningful inline comments to explain logic, especially in complex sections.
  - Each module or file must begin with a header comment stating:
    - Purpose of the file/module
    - Author and date
    - Version number
  - A complete **Developer Guide** or **README.md** must be included, covering:
    - Project setup instructions
    - Folder structure
    - Dependency list and installation commands
    - Configuration details (e.g., database, environment variables)

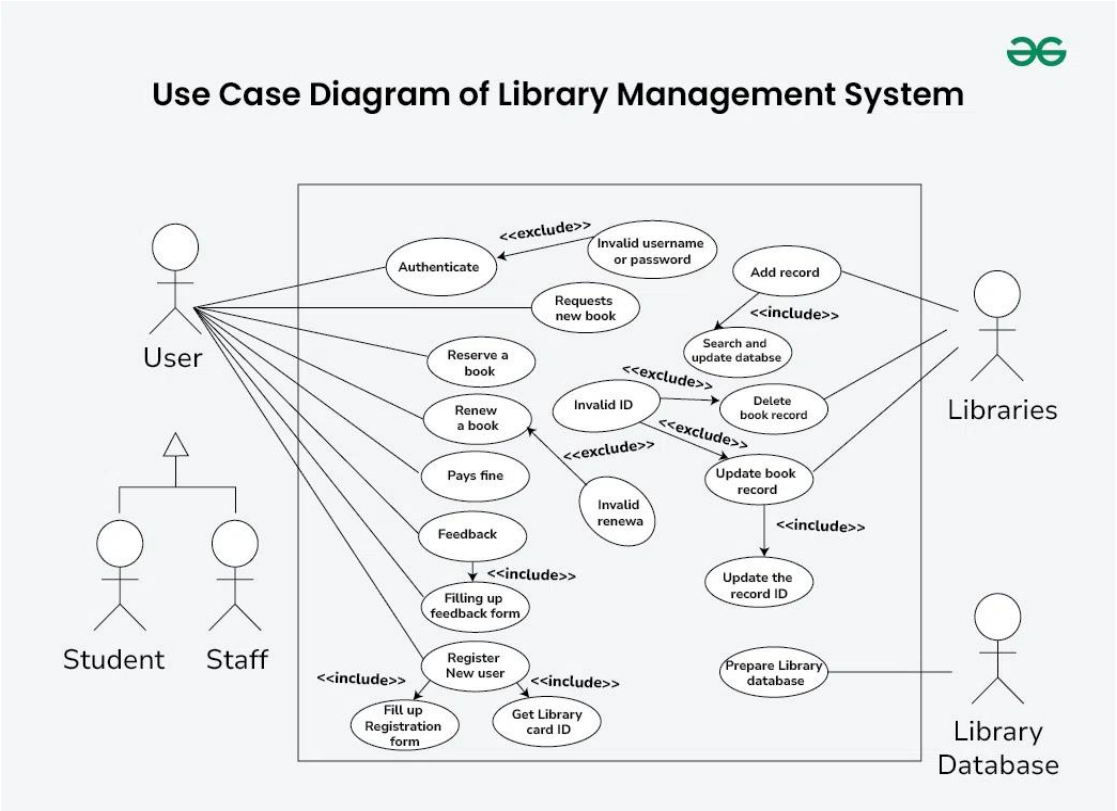
- **Version Control (Git):**
  - The LMS source code must be managed using **Git** to track changes, collaborate with multiple developers, and maintain a clean development history.
  - A standard Git branching model (e.g., main, dev, feature/*, bugfix/*) should be followed to isolate new features and hotfixes.
  - All changes must be committed with meaningful messages.
  - Each version release must be tagged (e.g., v1.0.0, v1.1.0) for easy rollback and historical reference.
  - Git repositories should be hosted on platforms such as:
    - GitHub
    - GitLab
    - Bitbucket (private or public depending on project policy)
- **Code Review and Collaboration:**
  - Code changes should be submitted via pull/merge requests and reviewed before being merged into the main branch.
  - Continuous integration (CI) pipelines may be integrated for testing and deployment automation (optional for advanced teams).

4.3 Disaster Recovery Preparedness (Optional Extension)

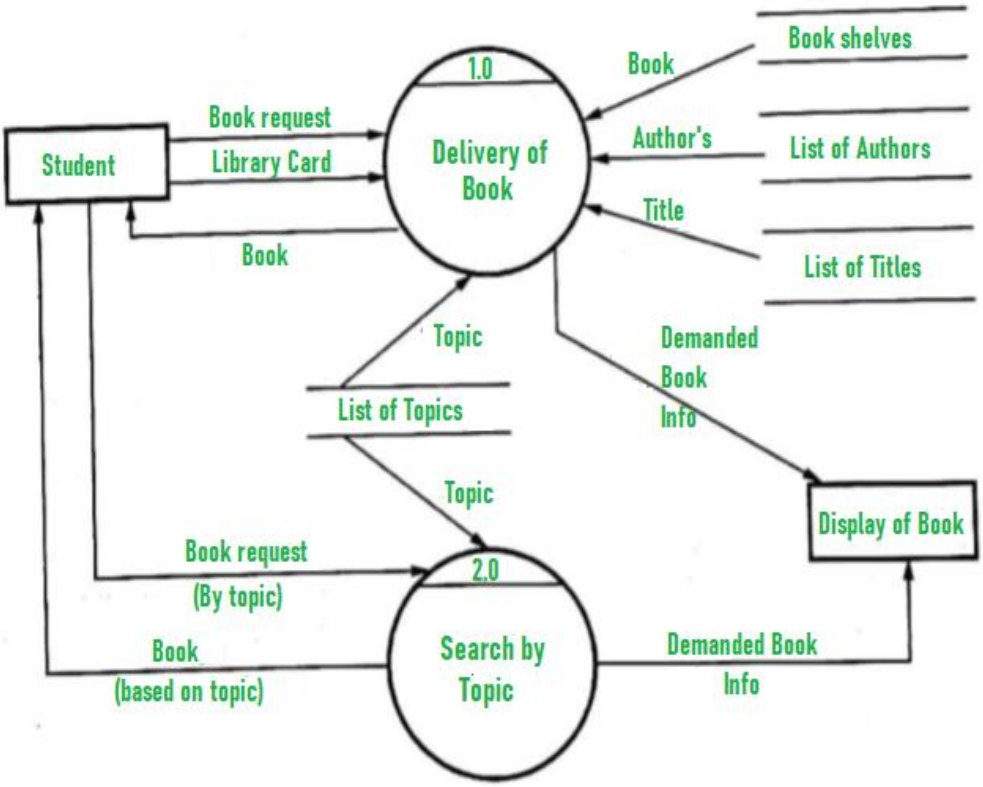
Although not mandatory, it is highly recommended to define disaster recovery protocols in combination with backups:

- A step-by-step recovery document should be maintained.
- A quarterly test of the recovery process should be performed to verify that backups can be restored successfully.

5. USECASE DIAGRAM



6. DATA FLOW DIAGRAM



Level 1 DFD