SRS Document (Software Requirement Specification)

What is a software requirements specification (SRS)?

SRS is a document that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill all stakeholders. (Business, user needs)

It describes the nature of a project, software or application. In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application.

1. Define the product's purpose
2. Describe what you're building
3. Detail requirements 4. Deliver it for approval.

Why use an SRS document?
An SRS gives you a complete picture of your entire project. It provides a single source of truth that every team involved in development will follow. It is your plan of action and keeps all your teams from development to maintenance on the same page.

SRS Document

Content

1. Introduction

    1.1 Purpose

    The purpose of the library management system is to improve access to information, enhance user experience, promote inclusivity, address sustainability concerns, and foster community engagement by leveraging online platforms to overcome the limitations of traditional physical libraries.

    1.2 Intended Audience and Reading Suggestions

    Intended Audience:

    Librarians and Library Staff

    Administrators

    IT Professionals

Library Patrons

Stakeholders

Reading Suggestions:

Librarians and Library Staff: Detailed operational procedures.

Administrators: Overview of purpose, scope, outcomes, and budget.

IT Professionals: Technical details on system architecture and security.

Library Patrons: User-focused features and benefits.

Stakeholders: Executive summary aligning with organizational goals.

1.3 Intended Use

The intended use of the library management system (LMS) is to efficiently manage library resources and services. This includes functions such as catalog management, user management, resource discovery, borrowing and circulation, reservation management, interlibrary loan, administrative tasks, reporting and analytics, integration with library services, and ensuring accessibility compliance. Overall, the LMS aims to enhance the accessibility, efficiency, and effectiveness of library operations for both users and staff.

1.4 Project Scope

- This encompasses the specific features and functionalities the LMS will offer, such as catalog management, user management, borrowing and circulation, reservation management, interlibrary loan, reporting, and administrative functions.
- This involves the technical specifications and infrastructure needed to develop, deploy, and maintain the LMS, including hardware, software, databases, programming languages, and development tools.
- This defines the design and usability aspects of the LMS, including the interface for library staff to manage the system and for users to access library services and resources.
- This outlines how the LMS will integrate with existing library systems, databases, and third-party services to ensure seamless operation and data interoperability.
- This addresses how data will be collected, stored, managed, and secured within the LMS, including data backup, retention, privacy, and compliance with relevant regulations.

- This specifies the capacity of the LMS to handle increasing volumes of users, transactions, and data over time, while maintaining acceptable levels of performance and responsiveness.
- This ensures that the LMS is accessible to users with disabilities, conforming to accessibility standards and providing features such as screen readers, keyboard navigation, and alternative text.
- This outlines the measures taken to secure the LMS against unauthorized access, data breaches, and cyber threats, including user authentication, encryption, access controls, and audit trails.
- This includes provisions for training library staff and users on how to use the LMS effectively, as well as ongoing technical support and troubleshooting services.
- This defines the project timeline, including key milestones, deliverables, dependencies, and deadlines for completing different phases of development, testing, and deployment.

1.5 Definition and Acronyms

- LMS: Library Management System
- ILS: Integrated Library System
- OPAC: Online Public Access Catalog
- RFID: Radio Frequency Identification
- API: Application Programming Interface
- SQL: Structured Query Language
- HTTPS: Hypertext Transfer Protocol Secure
- JSON: JavaScript Object Notation
- LDAP: Lightweight Directory Access Protocol
- XML: Extensible Markup Language
- OCR: Optical Character Recognition
- DOI: Digital Object Identifier
- ISBN: International Standard Book Number
- ISSN: International Standard Serial Number
- ADA: Americans with Disabilities Act
- GDPR: General Data Protection Regulation
- PCI DSS: Payment Card Industry Data Security Standard
- SLA: Service Level Agreement
- KPI: Key Performance Indicator

2. Overall Description

2.1 Product Perspective

The Product Perspective section of the Library Management System (LMS) document describes how the LMS interacts with existing library systems, interfaces with users and administrators, integrates with external services, identifies dependencies, and discusses its scalability and extensibility.

2.2 Product Features

The Product Features section of the Library Management System (LMS) document outlines the functionalities and capabilities offered by the system. Product features of an LMS include:

1. Catalog Management: Allows librarians to manage and organize library collections, including adding, editing, and deleting records for books, periodicals, multimedia items, and other resources.

2. User Management: Enables administrators to manage user accounts, including registration, authentication, profile management, and permission settings.

3. Borrowing and Circulation: Facilitates the borrowing, renewal, and return of library materials by users, with features such as check-in/check-out, due date reminders, and fine calculations.

4. Reservation Management: Allows users to reserve or place holds on items that are currently unavailable, with notifications sent when items become available for pickup.

5. Search and Discovery: Provides users with tools for searching, browsing, and discovering library resources based on various criteria such as title, author, subject, keywords, and metadata.

6. Interlibrary Loan: Enables users to request materials from other libraries or institutions through interlibrary loan services, with tracking and notification features to manage requests.

7. Reporting and Analytics: Generates reports and analytics on library usage, circulation statistics, collection trends, and other key metrics to inform decision-making and resource allocation.

8. Integration: Integrates with external systems and services such as online databases, digital repositories, authentication systems, and payment gateways to provide seamless access to resources and services.

9. Accessibility: Ensures that the system is accessible to users with disabilities, conforming to accessibility standards and providing features such as screen readers, keyboard navigation, and alternative text.

10. Security: Implements security measures to protect user data, prevent unauthorized access, and safeguard digital assets, including user authentication, encryption, access controls, and audit trails.

11. Customization and Configuration: Allows administrators to customize and configure the system to meet the specific needs and preferences of the library, including branding, workflows, and user interface settings.

12. Support and Maintenance: Provides technical support and maintenance services to ensure the ongoing operation, performance, and reliability of the system, including software updates, bug fixes, and troubleshooting assistance.

2.3 Operating Environment

- distributed database
- client/server system
- Operating system: Windows, Android, iOS
- database: SQL + database
- platform: python, Tensor flow Python

2.4 Design and Implementation Constraints

1. Technological Constraints: Limitations imposed by the technology stack chosen for the LMS, including compatibility issues, platform dependencies, and restrictions imposed by third-party libraries or frameworks.

2. Budgetary Constraints: Limitations on available resources, such as budget, time, and personnel, which may affect the scope, schedule, and quality of the LMS project.

3. Time Constraints: Deadlines and timeframes that must be adhered to for the completion of the project, including milestones, deliverables, and project schedules.

4. Scalability Constraints: Limitations on the scalability and performance of the system, including constraints on hardware resources, database capacity, and concurrent user access.

5. Regulatory Constraints: Compliance requirements imposed by laws, regulations, and standards relevant to the operation of the LMS, such as data protection regulations, accessibility standards, and copyright laws.

6. Integration Constraints: Limitations related to the integration of the LMS with existing systems, databases, and services, including compatibility issues, data format requirements, and API limitations.

7. User Requirements: Constraints imposed by user requirements and preferences, including usability constraints, accessibility requirements, and cultural or linguistic considerations.

8. Security Constraints: Limitations related to security requirements and considerations, including data encryption, access controls, authentication mechanisms, and compliance with security standards and best practices.

9. Legacy Systems: Constraints imposed by legacy systems or technologies that the LMS must interface with or replace, including data migration challenges, interoperability issues, and dependencies on outdated technologies.

10. Resource Constraints: Limitations on available resources, such as hardware, software, budget, and personnel, which may impact the design, development, and implementation of the LMS.

2.5 Assumptions and Dependencies

The Assumptions and Dependencies section of the Library Management System (LMS) document outlines the underlying assumptions made during the planning and development of the system, as well as the dependencies that may impact its implementation. Here are typical assumptions and dependencies:

Assumptions:

1. Availability of Resources: Assumes that necessary resources, such as funding, hardware, software, and personnel, will be available throughout the project lifecycle.

2. Stakeholder Cooperation: Assumes active participation and cooperation from stakeholders, including librarians, administrators, IT staff, and end-users, in defining requirements, providing feedback, and testing the system.

3. Regulatory Compliance: Assumes compliance with relevant laws, regulations, and standards, such as data protection regulations, accessibility standards, and copyright laws.

4. Technical Expertise: Assumes the availability of technical expertise and skills required for system design, development, testing, deployment, and maintenance.

5. User Acceptance: Assumes that users will accept and adopt the LMS, providing positive feedback and effectively utilizing the system to meet their needs.

6. System Integration: Assumes successful integration with existing library systems, databases, and services, without significant compatibility issues or data migration challenges.

7. Scalability and Performance: Assumes that the system will be able to scale to accommodate increasing volumes of users, transactions, and data, while maintaining acceptable levels of performance and responsiveness.

8. Security Measures: Assumes effective implementation of security measures, such as data encryption, access controls, and authentication mechanisms, to protect user data and ensure system integrity.

Dependencies:

1. Third-party Libraries and APIs: Dependencies on external libraries, frameworks, and APIs for implementing certain features or functionalities within the LMS.

2. Hardware and Software Dependencies: Dependencies on specific hardware components, operating systems, database management systems, programming languages, and development tools required for system development and deployment.

3. Data Dependencies: Dependencies on data sources, such as library catalogs, databases, digital repositories, and external services, for accessing and managing library resources within the LMS.

4. Integration Dependencies: Dependencies on the availability and compatibility of external systems, databases, and services that the LMS must integrate with, such as authentication systems, payment gateways, and digital content providers.

5. Vendor Dependencies: Dependencies on vendors or suppliers for delivering hardware, software licenses, technical support, and other services required for system implementation and operation.

3. System Features and requirements

1. Catalog Management:

- Ability to add, edit, and delete library resources.

- Support for various resource types such as books, journals, multimedia items, etc.
- Organizing resources into categories, genres, or subjects.

2. User Management:

- User registration and authentication.
- User profile management.
- Permission settings for different user roles (e.g., student, administrator).

3. Borrowing and Circulation:

- Check-in/check-out functionality.
- Managing loan periods and due dates.
- Renewal of borrowed items.
- Fine calculation and payment processing.

4. Reservation and Hold Management:

- Reservation of items that are currently unavailable.
- Notification when reserved items become available for pickup.

5. Search and Discovery:

- Advanced search capabilities (e.g., keyword search, title search, author search).
- Filters for refining search results (e.g., by publication year, format, language).
- Recommendation algorithms to suggest relevant resources.

6. Interlibrary Loan:

- Requesting materials from other libraries.
- Tracking and managing interlibrary loan requests.

7. Reporting and Analytics:

Generating reports on library usage, circulation statistics, etc.

Analytics for understanding user behavior and preferences.

8. Integration:

- Integration with external databases, digital repositories, and online resources.
- APIs for seamless integration with other library systems.

9. Accessibility:

- Ensuring accessibility features for users with disabilities (e.g., screen readers, keyboard navigation).
- Compliance with accessibility standards and regulations.

10. Security:

- Data encryption for sensitive information.
- Access controls to restrict unauthorized access.
- Authentication mechanisms (e.g., username/password, multi-factor authentication).

11. Customization and Configuration:

- Customizable user interface.
- Configuration options for system settings and preferences.

12. Support and Maintenance:

- Technical support for users and administrators.
- Regular updates and maintenance to ensure system reliability and security.

3.1 Functional Requirements

1. User Authentication and Authorization:
   - Users should be able to register and log in securely.
   - Different levels of access should be granted based on user roles (e.g., librarian, administrator, patron).

2. Catalog Management:
   - Ability to add, edit, and delete library resources (e.g., books, journals, multimedia items).
   - Organizing resources into categories or subjects.
   - Adding metadata such as title, author, publication year, and description.

3. Borrowing and Circulation:
    - Check-in/check-out functionality for borrowing library materials.
    - Managing loan periods and due dates.
    - Renewal of borrowed items.
    - Fine calculation and payment processing for overdue items.

4. Reservation and Hold Management:
    - Allowing users to reserve or place holds on items that are currently unavailable.
    - Notification when reserved items become available for pickup.

5. Search and Discovery:
    - Advanced search capabilities (e.g., keyword search, title search, author search).
    - Filters for refining search results (e.g., by format, publication year, language).
    - Recommendation algorithms to suggest relevant resources based on user preferences.

6. Interlibrary Loan:
    - Ability to request materials from other libraries.
    - Tracking and managing interlibrary loan requests.

7. User Profile Management:
    - Users should be able to update their profile information (e.g., contact details, preferences).
    - View borrowing history and fines accrued.

8. Reporting and Analytics:
    - Generating reports on library usage, circulation statistics, and inventory management.
    - Analytics for understanding user behavior and resource popularity.

9. System Administration:
    - Administrative interface for managing system settings, configurations, and user accounts.

- Adding and removing staff members, assigning roles and permissions.

10. Integration:
    - Integration with external databases, digital repositories, and online resources.
    - APIs for seamless integration with other library systems or third-party services.

4. External Interface Requirements

   4.1 User Interfaces

   4.2 Hardware Interfaces

   4.3 Software Interfaces

   4.4 Communication Interfaces

5. Non – Functional Requirements

   5.1 Usability Requirements

   - Intuitive user interface design for easy navigation and efficient use.
   - Consistent layout and design across different modules and screens.
   - Accessibility features to accommodate users with disabilities (e.g., screen readers, keyboard navigation).
   - User-friendly error messages and prompts for guidance.
   - Clear documentation and help resources for users and administrators.

   5.2 Performance Requirements

   - Fast response times for search queries, page loading, and transaction processing.
   - Scalability to handle increasing numbers of users, transactions, and data volumes.
   - Efficient resource utilization to optimize system performance and responsiveness.
   - Minimal downtime and high availability to ensure continuous access to library services.

5.3 Security Requirements

- Data encryption to protect sensitive information (e.g., user credentials, transaction data).
- Access controls and permission settings to restrict unauthorized access to system functionalities and resources.
- Secure authentication mechanisms (e.g., username/password, multi-factor authentication).
- Logging and auditing features to track user activities and system events for security monitoring.
- Compliance with relevant security standards and regulations (e.g., GDPR, PCI DSS).

5.4 Safety Requirements

- Backup and recovery procedures to safeguard against data loss and system failures.
- Disaster recovery plans to minimize downtime and ensure system continuity in case of emergencies or disasters.
- Redundancy and failover mechanisms to maintain service availability during hardware or network failures.
- Regular system backups and data replication to prevent data loss and facilitate recovery.

5.5 Software Quality Attributes

- Reliability: System should perform consistently and accurately without errors or unexpected behavior.
- Maintainability: Codebase should be well-organized, documented, and modular to facilitate future enhancements and maintenance.
- Portability: System should be deployable across different hardware platforms, operating systems, and environments.
- Testability: Code should be written with testing in mind to enable comprehensive test coverage and automation.
- Scalability: System should be able to accommodate growth in users, transactions, and data volumes without significant performance degradation.

6. Deliver for Approval

1. Document Review:
   - Distribute the completed LMS document to relevant stakeholders, including librarians, administrators, IT staff, and other decision-makers.
   - Schedule a meeting or review session to discuss the contents of the document and address any questions or concerns.

2. Feedback Collection:
   - Solicit feedback from stakeholders regarding the proposed features, requirements, assumptions, and constraints outlined in the document.
   - Document any feedback received and consider incorporating relevant suggestions or modifications into the final version of the document.

3. Approval Process:
   - Present the revised document to key stakeholders for approval.
   - Obtain formal approval or sign-off from relevant authorities, such as department heads, project sponsors, or executive management.

4. Documentation Retention:
   - Maintain records of the approved document and associated approvals for future reference.
   - Archive previous versions of the document and related communication for documentation purposes.

5. Communication Plan:
   - Communicate the approval status of the document to all stakeholders, including those involved in the project and any impacted parties.
   - Ensure that stakeholders are informed of any changes or updates to the project plan based on the approved document.

6. Next Steps:
   - Once approval is obtained, proceed with the next steps in the project lifecycle, such as system design, development, testing, and deployment.
   - Update project documentation and plans as necessary to reflect approved requirements and decisions.

2. Define your Product's Purpose

This introduction is very important as it sets expectations that we will hit throughout the SRS. Some items to keep in mind when defining this purpose include:

Intended Audience and Intended Use

Define who in your organization will have access to the SRS and how they should use it. This may include developers, testers, and project managers. It could also include stakeholders in other departments, including leadership teams, sales, and marketing. Defining this now will lead to less work in the future.

Product Scope

What are the benefits, objectives, and goals we intend to have for this product? This should relate to overall business goals, especially if teams outside of development will have access to the SRS.

Definitions and Acronyms

It's important to define the risks in the project. What could go wrong? How do me mitigate these risks? Who is in charge of these risk items?
For example, if the failure of a medical device would cause slight injury, that is one level of risk. Taking into account the occurrence level and the severity, we can come up with a strategy to mitigate this risk.

3. Describe What You Will Build

Your next step is to give a description of what you're going to build. Is it a new product? Is it an add-on to a product you've already created? Is this going to integrate with another product? Why is this needed? Who is it for?

Understanding these questions on the front end makes creating the product much easier for all involved.

User Needs

Describe who will use the product and how. Understanding the user of the product and their needs is a critical part of the process.

Who will be using the product? Are they a primary or secondary user? Do you need to know about the purchaser of the product as well as the end user? In medical devices, you will also need to know the needs of the patient.

Assumptions and Dependencies

What are we assuming will be true? Understating and laying out these assumptions ahead of time will help with headaches later. Are we assuming current technology? Are we basing this on a Windows framework? We need to take stock of these assumptions to better understand when our product would fail or not operate perfectly.

Finally, you should note if your project is dependent on any external factors. Are we reusing a bit of software from a previous project? This new project would then depend on that operating correctly and should be included.

4. Detail Your Specific Requirements

In order for your development team to meet the requirements properly, we MUST include as much detail as possible. This can feel overwhelming but becomes easier as you break down your requirements into categories. Some common categories are:

Functional Requirements

Functional requirements are essential to your product because, as they state, they provide some sort of functionality.

Asking yourself the question “does this add to my tool’s functionality?” Or “What function does this provide?” can help with this process. Within Medical devices especially, these functional requirements may have a subset of risks and requirements.

You may also have requirements that outline how your software will interact with other tools, which brings us to external interface requirements.

External Interface Requirements

External interface requirements are specific types of functional requirements. These are especially important when working with embedded systems. They outline how your product will interface with other components.

There are several types of interfaces you may have requirements for, including:

- User

- Hardware
- Software
- Communications

System Features

System features are types of functional requirements. These are features that are required in order for a system to function.

Other Non-functional Requirements

Non-functional requirements can be just as important as functional ones.

These include:

- Performance
- Safety
- Security
- Quality

The importance of this type of requirement may vary depending on your industry. In the medical device industry, there are often regulations that require the tracking and accounting of safety.

5. Deliver for Approval

We made it! After completing the SRS, you'll need to get it approved by key stakeholders. This will require everyone to review the latest version of the document.