Software Requirements Specification

for

Library Management System

Version 1.0 approved

Documentation Prepared by Khalid Tarek

Project Made by The RAD Squad

For Software Engineering (2) Final Project - EELU

26 November, 2020

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Intended Audience and Reading Suggestions 1

1.3 Project Scope 1

1.4 References 1

2. Overall Description 2

2.1 Product Features 2

2.1.1 User Authentication 2

2.1.2 Library Member Functionality 2

2.1.3 Librarian Functionality 2

2.1.4 Book Author Functionality 2

2.2 User Classes and Characteristics 2

2.3 Operating Environment 2

2.4 Design and Implementation Constraints 2

2.5 Assumptions and Dependencies 3

3. System Features 3

3.1 User Authentication 3

3.2 Library Member Functionality 4

3.3 Librarian Functionality 4

3.4 Book Author Functionality 5

4. External Interface Requirements 5

4.1 User Interfaces 5

4.2 Hardware Interfaces 6

4.3 Software Interfaces 6

4.4 Communications Interfaces 6

5. Other Nonfunctional Requirements 6

5.1 Performance Requirements 6

5.2 Safety Requirements 6

5.3 Security Requirements 7

5.4 Software Quality Attributes 7

6. Other Requirements 7

Appendix A: Glossary 7

Appendix B: Analysis Models 7

Appendix C: Issues List 7

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Khalid | 26/11/20 | Document Creation | 1.0 |
|  |  |  |  |

# Introduction

## Purpose

This simple software is intended to simulate an imaginary library system where library members, librarians, and authors can interact with a database holding their user accounts and their respective books through a minimalistic user interface. The main purpose of this project is to practice software engineering concepts and to act as the final project for the SWE305 - Software Engineering (2) course (Egyptian E-learning University) of fall 2020-2021.

## Intended Audience and Reading Suggestions

The intended readers for this document first and foremost are the coordinators of the SWE305 course. Secondly, this document is also intended for the developers and/or users of this system, to understand the different aspects of it and how interfaces communicate with the database described.

The first section gives a brief introduction about the software, followed by a section explaining targeted users and the environment on which this system will operate will be given. Then an overview of the software’s features and its requirements will be explained in more detail, followed by the interfaces with which the targeted user will be interacting with. Finally, the nonfunctional requirements of the system will be explained, followed by a list of appendices for this document.

## Project Scope

This project’s scope encompasses the very basic functionality of a library.

* This program allows members to borrow books, add credit with which they can pay the fines that generate when the user borrows a book.
* It also allows authors to add or remove their books (while this varies greatly in reality, since the event of releasing books to the public goes through many phases, including publishing as the like, but for the purposes of this software, it was dumbed down to authors adding their books directly).
* This software also gives librarians the ability to act as an “administrator” of a sort for the system, allowing them to create new member or author accounts or remove members, authors or books from the system. In reality though, the true administrators of the system will have total control of everything, but that’s beyond the scope of this system and will not be implemented other than in a conceptual view of the business use-case model.

This software is developed purely in Java, including the user interface and the backend interface for the database.

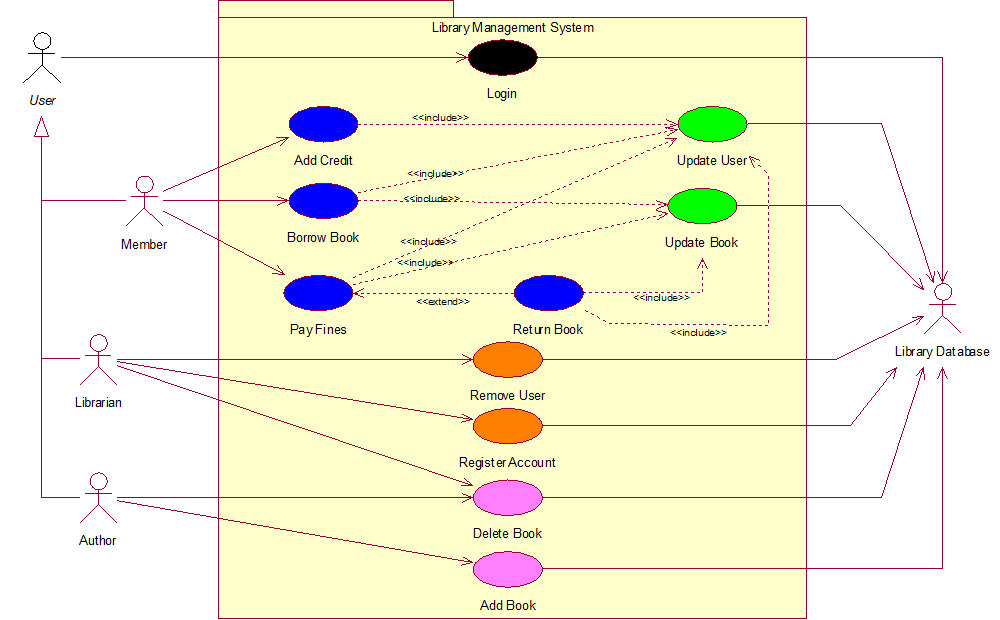
The database system used is further explained later in this document

## References

* [Karl E. Wiegers Software Requirements Specification Template](https://cs.gmu.edu/~dfleck/classes/cs421/spring08/srs_template.doc)
* UML Models.mdl (In the same directory as this document)

# Overall Description

## Product Features



**Figure 1:** Use Case Diagram

### User Authentication

The system can distinguish each user by their account credentials, i.e. username and password. This feature is noted in Figure 1 as the black colored use case by the generalized actor *User*.

### Library Member Functionality

The system can help members borrow books, track their borrowed books, keep credit, and allow fine payments for the borrowed books. This feature is noted in Figure 1 as all the green colored used cases by the actor *Member*.

### Librarian Functionality

The system can allow librarians to create new member or author accounts, remove these accounts and remove book entries. This feature is noted in Figure 1 as all the orange colored used cases by the actor *Librarian*, and one of the prominently *Author* use-case “Delete Book”.

### Book Author Functionality

The system can allow authors to add their books into the database or remove them. This feature is noted in Figure 1 as all the pink colored used cases by the actor *Author*.

## User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy.>

## Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

## Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

## Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## User Authentication

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

3.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

3.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

3.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Library Member Functionality

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

3.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

3.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

3.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Librarian Functionality

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

3.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

3.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

3.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Book Author Functionality

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

3.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

3.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

3.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

# External Interface Requirements

## User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

## Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

## Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

# Other Requirements

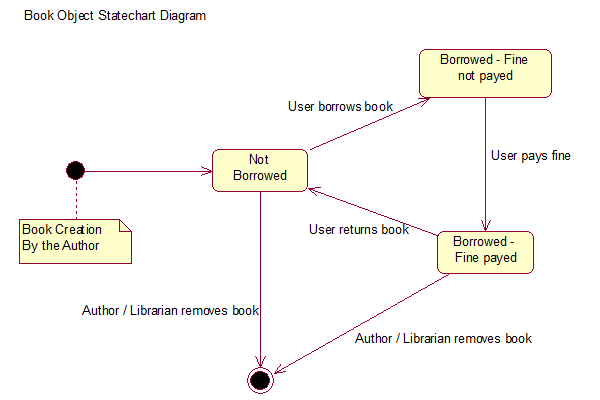
<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

* **User**:
* **Member**:
* **Librarian**:
* **Author**:
* **Book**:

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>



**Appendix Figure 1**: Book Object state-transition diagram

Appendix C: Issues List

< This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.>

**TBD**