Software Requirements Specification

Bibli.io

Prepared by

Joshua Butler	40045825	butler.joshua355@gmail.com
Mohammad Naimur Rashid	40027867	rashidn96@gmail.com
Suruthi Raju	40084709	suruthi25@gmail.com
Razine Ahmed Bensari	40029076	bensaria97@gmail.com
Nicolas Samaha	40027961	nicktherealog@gmail.com
Vincent Cerri	40034135	vincent.cerri@gmail.com
Justin Carrier	40026363	justind.carrier@gmail.com

Instructor: Dr. Constantine Constantinos

Course: Software Process (SOEN 341)

Date: 23 November 2018

Table of contents

List of figures

1. Introduction

Bibli.io is a software system that allows the library's users and administrators to perform business transactions. This Software Requirements Specification Document helps with highlighting the necessary features of the software project as gathered by the stakeholders. This document gives the overall detail of how the library system is going to perform.

1.1 Purpose

The purpose of this SRS document is to give a detailed description of the requirements for the Bibli.io software. It also accumulates and analyzes various idea that has mentioned in the project requirement document and also with respect to the User. Briefly, the purpose of the document is to provide a definitive overview of our library software product, its goal and parameters.

We have also provided information describing the project's description and specific requirements that were taken into consideration during the design phase. The intended audience for the document is aimed towards a technical audience that is familiar with UML design principles and software engineering. The document is to be used mainly for reference to the design rationale that was used during the inception of the software system to understand why certain design decisions were made. It will also justify system constraints, interface and interactions with other external applications.

1.2 Scope

The "Bibli.io" is a E-library Desktop Application(OPAC) which helps users and administrator to perform library related action. User can remotely access to library catalog to get the details of the item they desire. User can also Loan or return an item using Bibli.io. Meanwhile Library Administrator can also perform Admin related actions to be same application. Administrator can add/edit/remove an item from the library catalog for User interaction.

1.3 Definitions, acronyms, and abbreviations

Terms	Definitions
SRS	Software Requirement Specifications
UML	Unified Modeling Language
User	Someone who browses the library catalog
Administrator	Administrator who is given access to manage and control the library catalog
OPAC	Online Public Access Catalog
DESC	Description

1.4 References

The references are:

- Lecture Slides and Material provided by Dr. Constantine Constantinos.
- IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998.
- Library Terms - http://libguides.usc.edu/libraryterms

1.5 Overview:

The document includes four chapters. The following chapter, chapter 2, contains the overall system functionality and interaction with the external system. And, the third, provides the detailed functional requirements of the whole system given in the project description. The fourth and final chapter explains the detailed view of some of the important functional use cases that are covered in the SRS, SAD, and UML diagrams.

2. Overall description

This section gives the overview of whole “bibli.io” system. This section explains the basic system functionality, its interface with other system and the type of stakeholders. And, atlast, the constraints, assumptions, and dependencies.

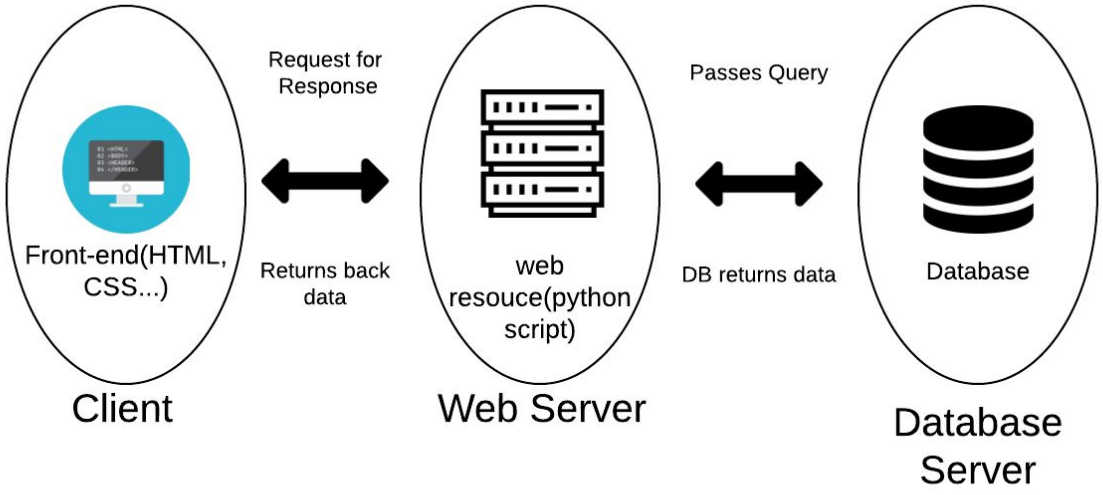

Figure 1. A client-server-database architecture

2.1 Product perspective

The system consists of three parts: one client interaction(front-end), one web portal and one database server. Our System Bibli.io will interacts with user for viewing library catalogs and searching the catalog along with admin privilege. The product is self-contained. The web client, server, and database needed to use the software system is ran on a single machine for the purpose of this academic project.

The UI(front-end) development is done with basic HTML and CSS, which interacts with the user and administrator. Based on the user needs the web resource(in our project Python Script) is used to communicate with the Database through SQL Queries to change the tables in

the database. This application uses the database to get data while the web portal will also add,modify and delete data.

With the Bibli.io application, the users will be able to search for items like Books, Magazines, Movies and Musics in the database. The result will be based on the criteria the user inputs. The administrator of the library system manages the catalog of the library. They can add, edit or delete an item in the library. They can also provide access to new user who can be another administrator.

2.2 User characteristics

There are two types of users that interact with Bibli.io system: users of Bibli.io application and administrators. Each types of users has different functionality and use to the system.

The desktop application of Bibli.io users can only use the application to view the library catalog and search the catalog. In addition he can loan the item and return them back with this application. In search operation for the user, there are multiple criteria the users can specify and all results matches all of those.

The administrators also can interact with desktop version of Bibli.io They manage the overall library catalog so there is no incorrect information within it. The administrator can manage the information for each and every item present in the library.

Due to the nature of the project being academic, the user interface has not been designed to be user-friendly; therefore, the software is aimed towards a more technical audience.

2.3 Constraints

There are many constraint that have to be considered when the project enters production, such as:

- The Internet connection. Example: The application fetches/updates/deletes data from the database over the Internet.

- The capacity of database. Since the database is used by many people at the same time, It may be forced to performance deterioration and to queue incoming requests and thus increasing the time to interact with data.

2.4 Assumptions and dependencies

The software system is a web application. Only a web browser and a connection to the server is needed to use the software system. The web browser is only desktop friendly.

3. Specific requirements

This section contains all requirements in detail. That is the Functional requirements (quality attributes). The quality attributes are listed according to the *ISO/IEC 25010* standard that classifies software quality in a structured set of characteristics and sub-characteristics.

3.1 External interfaces

A detailed description of all inputs and outputs in and out of the system. There are different types of interface that connects with a system namely: Hardware, Software, User and Communication Interface. In our Bibli.io system, we use below interface:

- Hardware: There is no such interface present in our system.
- Software: Communication between the database and the web portal consists of operation concerning reading, modifying and deleting the data.

User - User interaction for the software shall be compatible to any browser such as Internet Explorer, Mozilla and Chrome.

- Communication - HTTP Protocol for communication over the Internet.

3.2 Functional requirements

Functional requirements capture the intended behaviour of the system. This section gives all the functional requirement specified in the project description document. Below is the list for Bibli.io system

3.2.1 User Class 1 - The User

3.2.1.1 Functional requirement 1.1

TITLE: User registration

DESC: Given that a user has navigated to Bibli.io system. The user has to register themself to login to the library system. The user must provide first name, last name, phone, address username, password and email address to register.

3.2.1.2 Functional requirement 1.2

TITLE: User Login

DESC: Given that a user has registered to the system. User can give username and password to log into the system.

3.2.1.3 Functional requirement 1.3

TITLE: View Catalog

DESC: Once the user logs in to the system. They can view the item. They can also click the logo of system “Welcome to bibli.io” to view the catalog.

3.2.1.4 Functional requirement 1.4

TITLE: Search Catalog

DESC: Once the user logs in to the system. They can search for the item required by clicking on search button in the menu bar. They can search on specific item like Book, Magazine, Movies or Music by selecting from the drop down. They can search specific item by given field values.

3.2.1.5 Functional requirement 1.5

TITLE: Make Loan

DESC: Once the user logs in to the system. They can loan for the required item by giving item id to the system.

3.2.1.6 Functional requirement 1.6

TITLE: Sign Out

DESC: Once the user logs in to the system. They can leave the system by clicking on sign out button in the menu bar.

3.2.2 User Class 2 - The Administrator

3.2.2.1 Functional requirement 2.1

TITLE: Administrator Login

DESC: Given that an Administrator has registered to the system by default in the database or another administrator has made them as admin. Admin can give username and password to log into the system.

3.2.2.2 Functional requirement 2.2

TITLE: View Catalog

DESC: Once the Admin logs in to the system. They can view the item. They can also click the logo of system “Welcome to bibli.io” to view the catalog.

3.2.2.3 Functional requirement 2.3

TITLE: Search Catalog

DESC: Once the Admin logs in to the system. They can search for the item required by clicking on search button in the menu bar. They can search on specific item like Book, Magazine, Movies or Music by selecting from the drop down. They can search specific item by given field values.

3.2.2.4 Functional requirement 2.4

TITLE: Add Item

DESC: Once the Admin logs in to the system. They can add an item like Book, Magazine, Movies or Music by selecting from the drop down. They can give the details of the item as show in the fields to enter the item in the database.

3.2.2.5 Functional requirement 2.5

TITLE: Edit Item

DESC: Once the Admin logs in to the system. They can edit an item like Book, Magazine, Movies or Music by giving the item id in the field of the system. Once the item id is given the field for the item is populated. The data can correct and clicked on save to save into database.

3.2.2.6 Functional requirement 2.6

TITLE: Delete Item

DESC: Once the Admin logs in to the system. They can delete an item like Book, Magazine, Movies or Music by giving the item id in the field of the system. The data will be delete from the database

3.2.2.7 Functional requirement 2.7

TITLE: Process Return

DESC: Once the Admin logs in to the system. They can process return from the user. The item id of the loaned item has to be entered in the system so that the item is returned to the library system.

3.2.2.8 Functional requirement 2.8

TITLE: Register Admin

DESC: Once the Admin logs in to the system. They can make other potential person as admin by giving their details in the register Admin page so that the new person is registered in database and system.

3.2.2.9 Functional requirement 2.9

TITLE: View Active User Count

DESC: Once the Admin logs in to the system. They can view the active user that is accessing the system by clicking on View Active Count button.

3.2.2.10 Functional requirement 2.10

TITLE: Sign Out

DESC: Once the Admin logs in to the system. They can leave the system by clicking on sign out button in the menu bar.

3.3 Actor goal list

Actor	Goal
Administrator	Process Item(Add/Edit/Delete)
Client	Process Loan
Administrator/Client	View Catalog
Administrator/Client	Search Catalog

3.4 Use Case view

The use case model is shown in Figure 3 below.

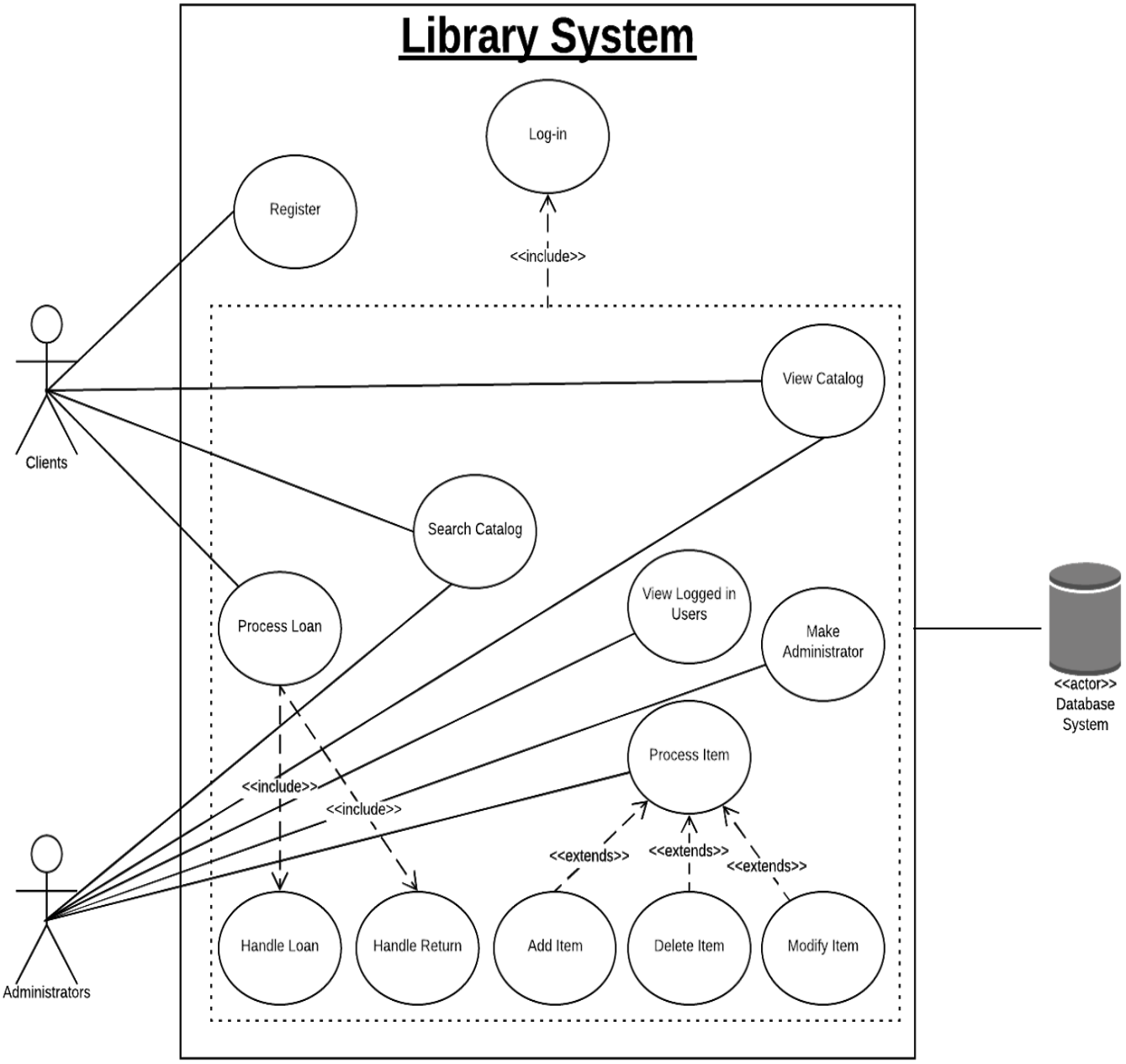

Figure 3. Use case model

4. Detailed View of Specific Use Cases

4.1 Importance/Prioritization of Use Cases

During requirements analysis, there are many factors that affect the importance of prioritizing use cases such as:

- Customer Priority
- Risk eg: performance, availability and suitability
- Complexity eg: Complexity of a phase
- Dependency eg: architectural impact like extends, modifies
- Other tactical constraints eg: demo to end user …

In our scenario, since we are doing a academic project in small scale, the Risk, Complexity, Dependency is not taken into consideration. We are focusing on the main scenarios that meet stakeholders expectations. The critical use cases to be developed are listed below:

Use Case: UC1 - Process item

Actor: Administrator

Pre-condition: Administrator is identified and authenticated

Post-condition: Entered insertions, deletions, and modifications of books, magazines, movies, and musics are stored in their respective catalogs.

Success scenario:

1. Administrator starts a session from a computer

2. Administrator enters required information for a print or media item

3. System validates and stores the information and confirms of the success of the operation

4. Administrator repeats step 2-3 until indicates done.

Extensions:

3a. If data is invalid, then system informs administrator of the error.

Use Case: UC2 – Process loan

Actor: Client

Pre-condition: Client is identified and authenticated

Post-condition: Entered loans and returns are stored in system

Success Scenario:

1. Client starts a session from a computer

2. Client enters required information for a print or media item to be loaned or returned

3. System validates and stores the information and confirms of the success of the operation

4. Client repeats step 2-3 until indicates done.

Extensions:

3a. If data is invalid, then system informs client of the error.

Use Case: UC3 – View Catalog

Actor: User

Pre-condition: User is identified and authenticated

Post-condition: User obtains a view of all catalog items

Success Scenario:

1. User starts a session from a computer

2. User requests to view a catalog

3. System fetches and displays catalog information

Use Case: UC4 – Search Catalog

Actor: User

Pre-condition: User is identified and authenticated

Post-condition: User finds searched record

Success Scenario:

1. User starts a session from a computer

2. User enters required information for a print or media item

3. System validates and returns the information of the search result operation

Extensions:

3a. If search is unsuccessful, then system informs user of no record found.

Use Case: UC5 – Register

Actor: Client

Pre-condition: N/A

Post-condition: Client is registered

Success Scenario:

1. Client starts a session from a computer

2. Client enters required information for registration

3. System validates and stores the information and confirms of the success of the operation

Extensions:

3a. If data is invalid, then system informs client of the error.

Use Case: UC6 – Make Administrator

Actor: Administrator

Pre-condition: Administrator is identified and authenticated

Post-condition: A new administrator is created in the system

Success scenario:

1. Administrator starts a session from a computer

2. Administrator enters required information for new administrator

3. System validates and stores the information and confirms of the success of the operation

4. Administrator repeats step 2-3 until indicates done.

Extensions:

3a. If data is invalid, then system informs administrator of the error.

4.2 Analysis Models

This section provides the sequence diagram, domain model and system operator for the important use cases mentioned above.

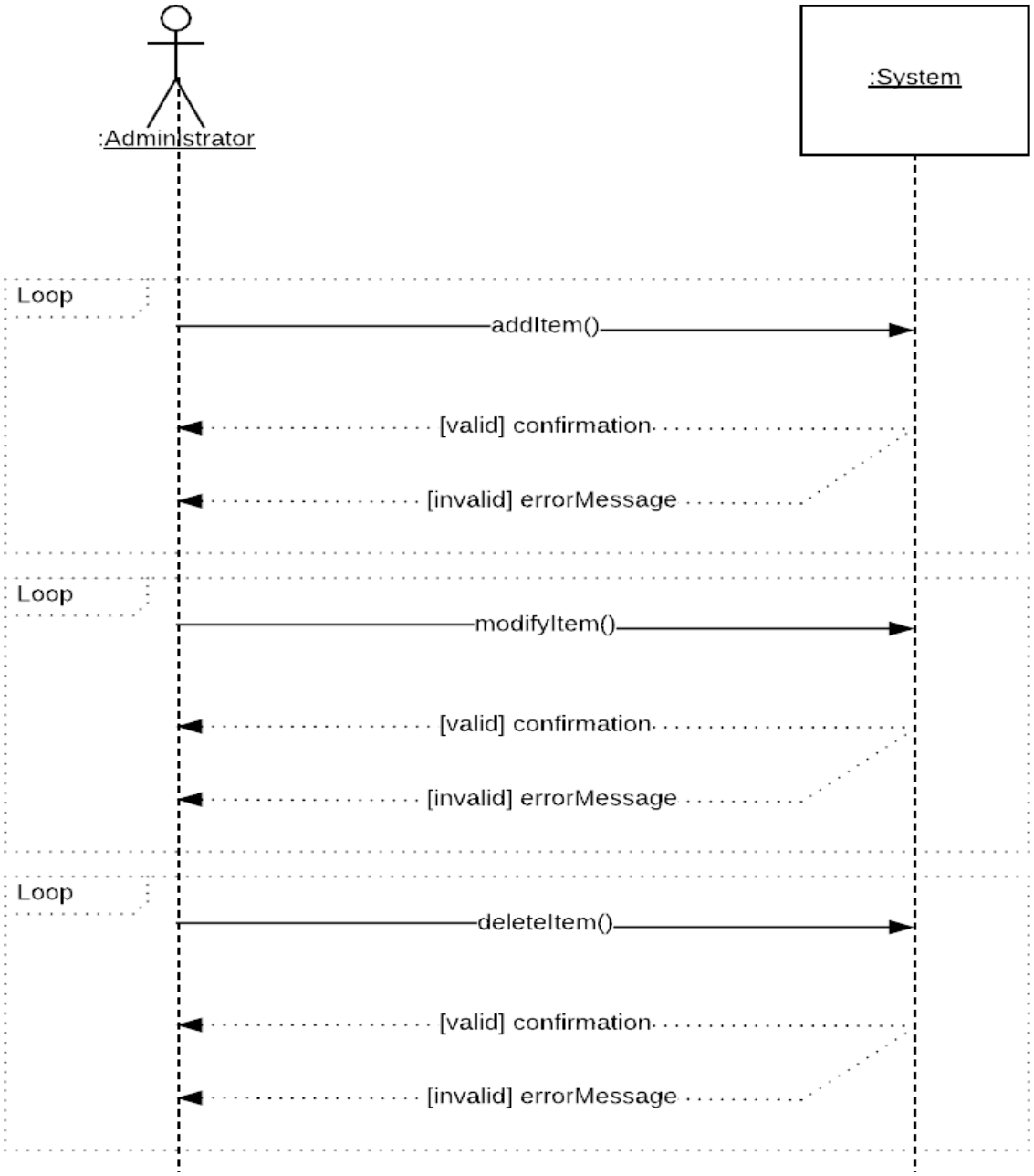

Figure 3. UC1 SSD

Contract CO1: addItem
Operation: addItem(spec)
Cross References: Use Case: Add Item.
Preconditions: An entry session is present..
Postconditions:

1. An instance of item i was created. (instance creation)
2. i was associated with ItemCatalog. (association formed)
3. i attributes defined with keyworded arguments from spec list. (attribute modification)

Contract CO2: modifyItem
Operation: modifyItem(spec)
Cross References: Use Case: Modify Item.
Preconditions: An entry session is present.
Postconditions:

1. An instance of item i was retrived from ItemCatalogue based specifications provided. (association formed)
2. i attributes updated with appropriate arguments from spec list. (attribute modification)

Contract CO3: deleteItem
Operation: deleteItem(id)
Cross References: Use Case: Delete Item.
Preconditions: An entry session is present.
Postconditions:

1. An instance of item i was removed from ItemCatalogue based specifications provided. (association broken)

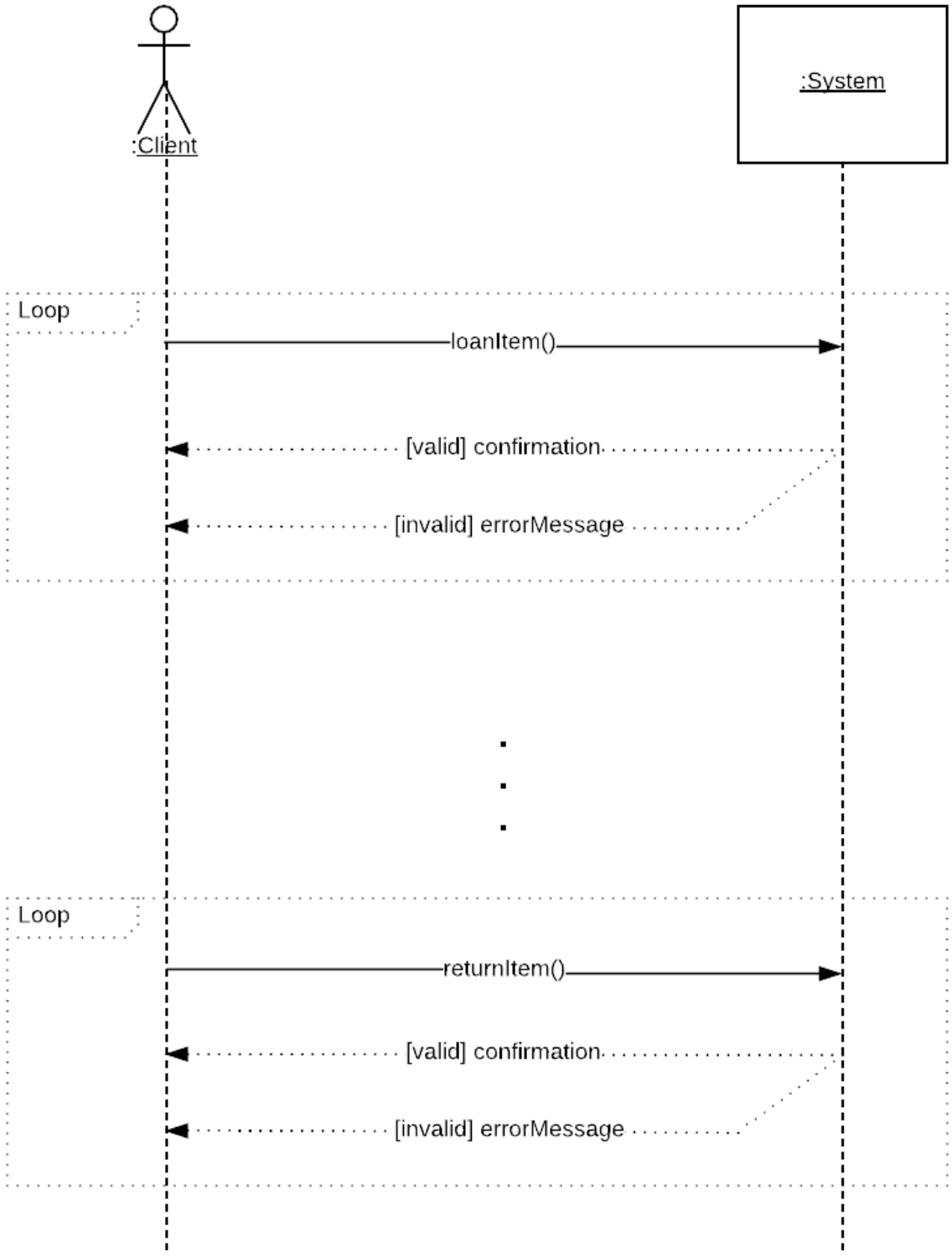

Figure 4. UC2 SSD.

Contract CO4: loanItem
Operation: loanItem()
Cross References: Use Case: Handle Loan.
Preconditions: Loan session is present.
Postconditions:

1. An instance of loan item li was retrived from ItemCatalogue based specifications provided. (association formed)
2. li.loaned was set to True. (attribute modification)

Contract CO5: returnItem
Operation: returnItem(spec)
Cross References: Use Case: Handle Return.
Preconditions: Return session is present.
Postconditions:

1. An instance of return item ri was retrived from Catalogue based specifications provided. (association formed)
2. ri.loaned was set to False. (attribute modification)

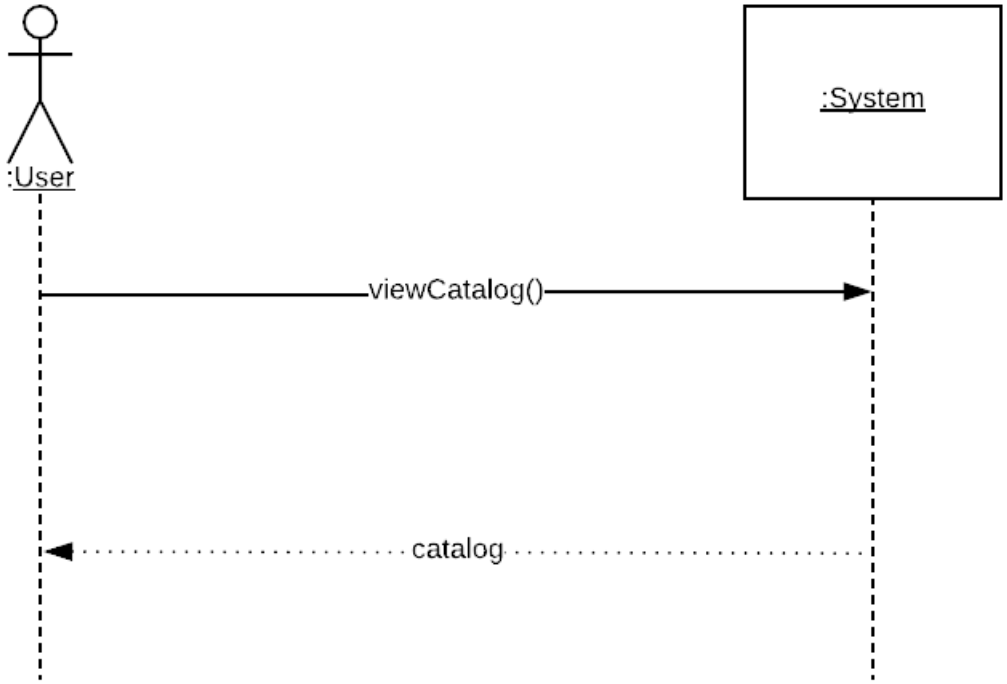

Figure 5. UC3 SSD

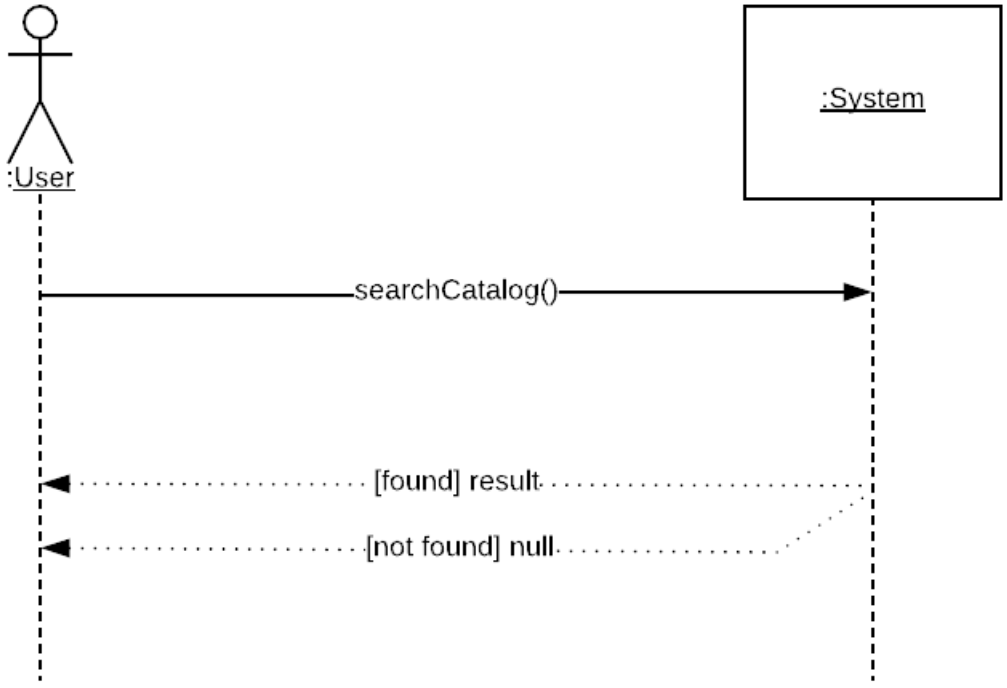

Figure 6. UC4 SSD

Contract CO6: viewCatalog
Operation: viewCatalog()
Cross References: Use Case: View Catalog.
Preconditions: Client is authenticated.
Postconditions:

1. Catalog was associated with View (association formed)

Contract CO7: searchCatalog
Operation: searchCatalog()
Cross References: Use Case: Search Catalog.
Preconditions: Client is authenticated.
Postconditions:

1. An instance of Search Catalog sc was created (instance creation).
2. sc was associated with Catalog (association formed).
3. sc.items is set to a list of elements from Catalog with matching provided specifications . (attribute modification).
4. sc was associated with View (association formed).

System Operations:

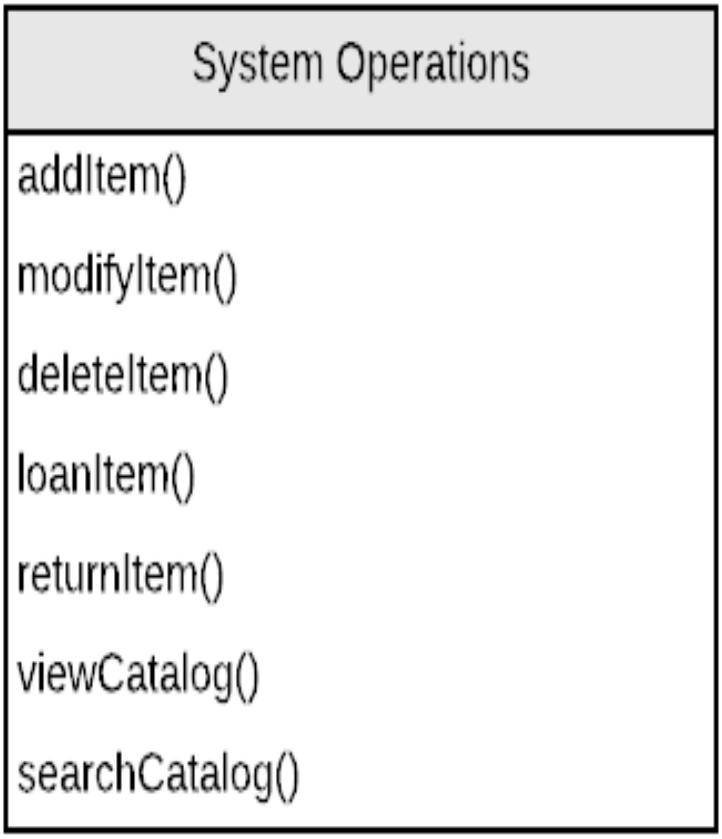

Figure 7. Systems Operations

Domain Model:

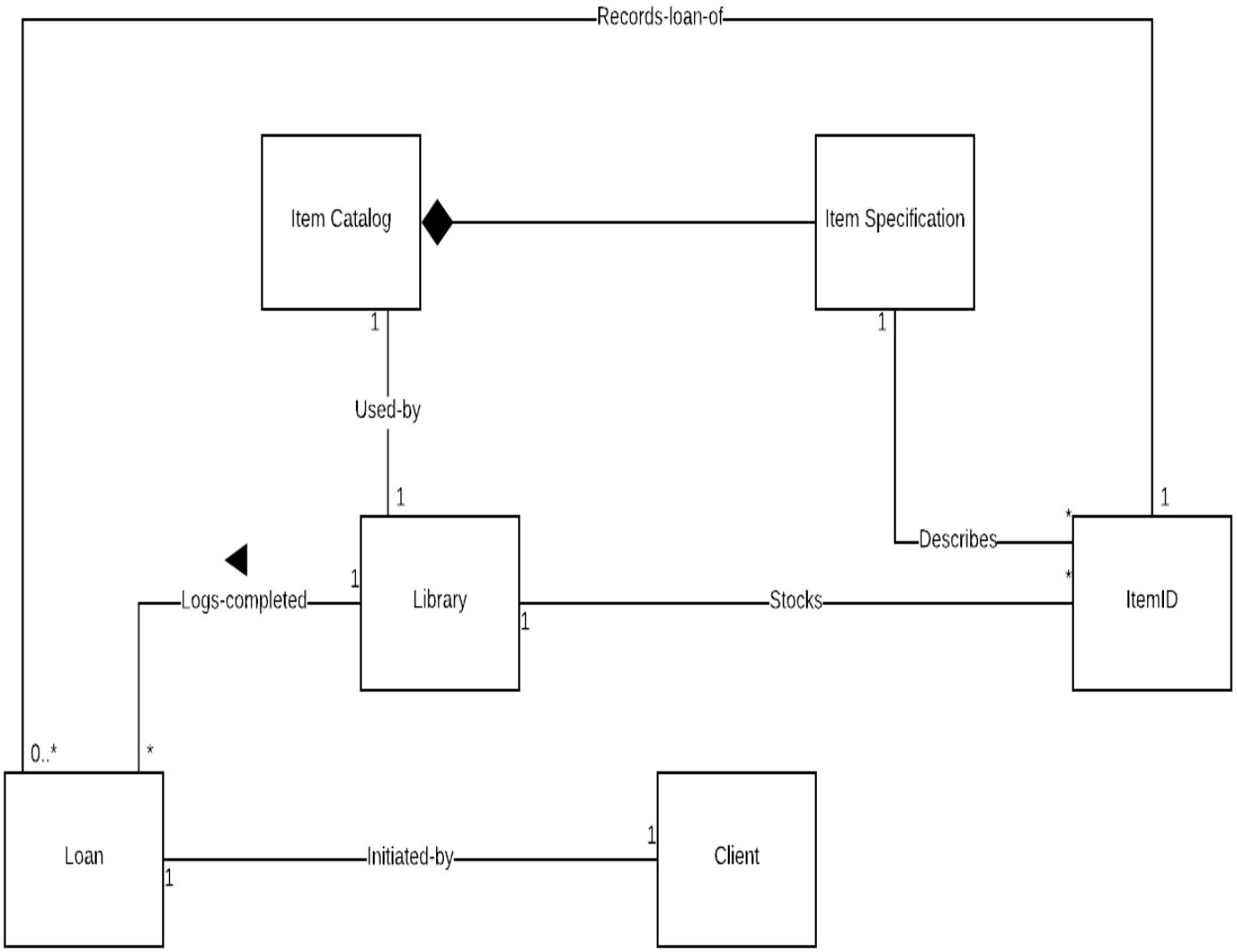

Figure 8. Domain Model

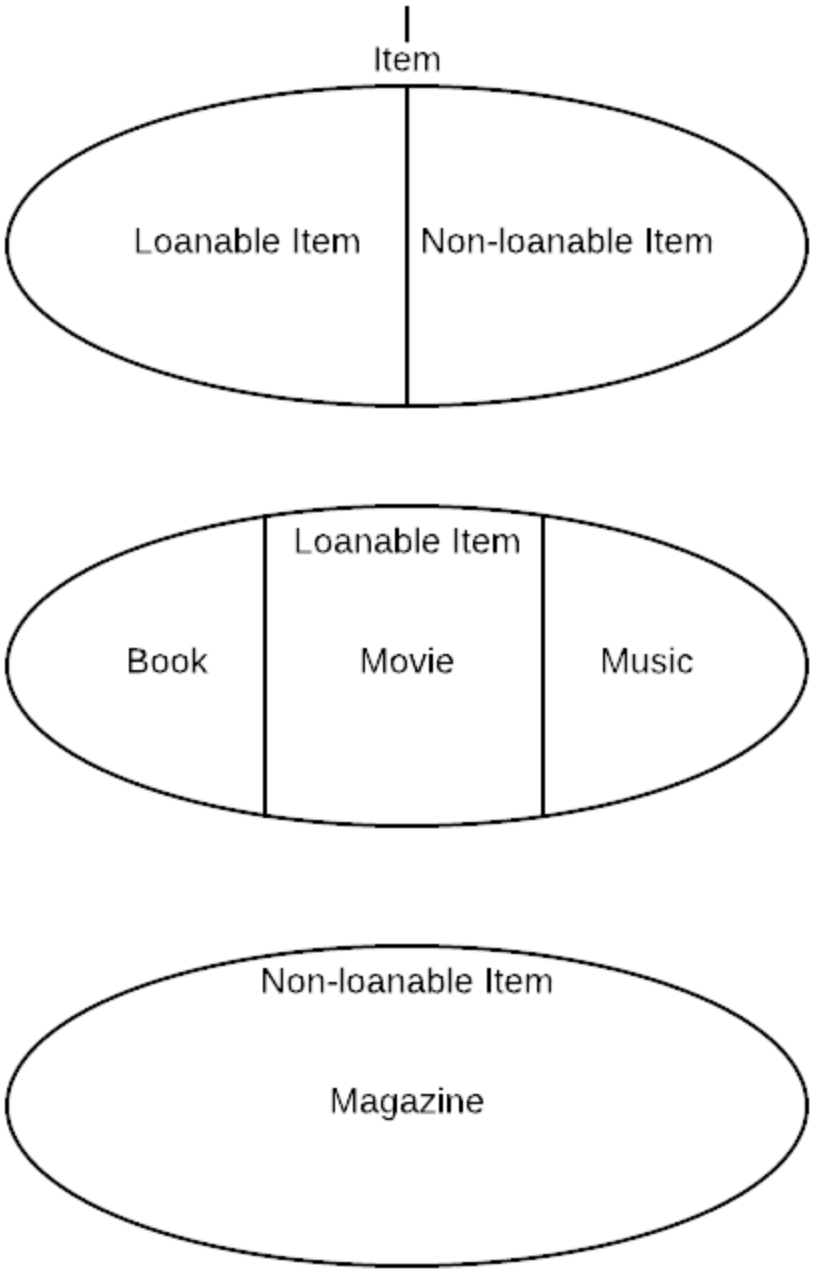

Figure 9. Domain Model Generalizations