Software Requirements Specification

Version 7.0
for
Biblio

Prepared by

Name	Student ID	Email
Abdulla Alhaj Zin	40013496	abdullah.hzen@gmail.com
Dave Bhardwaj	40000679	davebhardwaj1@gmail.com
Giovanni Gebran	40018637	ggebran95@gmail.com
Kayne Herrmann	40007153	kh_group@outlook.com
Kenza Boulisfane	40043521	kenza.boulisfane@gmail.com
Kevin Camellini	26771009	kevincamellini@gmail.com
Kevin Lin	40002383	Lin_Kevin_1995@hotmail.com
Kevin Yau	27058276	kevin.yau@outlook.com
Nizar Belhassan	27519443	M.nizar.belhassan@gmail.com
Nour El Natour	40013102	nour_elnatour@hotmail.com

Instructor: Dr. C. Constantinides

Course: Software Architecture and Design I

Date: 11/23/2018

Document history

Date	Version	Description	Author
09/19/2018	<1.0>	SRS 1.0	Team 5
09/26/2018	<1.5>	SRS 1.5	Team 5
09/28/2018	<2.0>	SRS 2.0	Team 5
09/29/2018	<2.5>	SRS 2.5	Team 5
10/10/2018	<3.0>	SRS 3.0	Team 5
10/12/2018	<3.5>	SRS 3.5	Team 5
10/21/2018	<4.0>	SRS 4.0	Team 5
10/22/2018	<4.5>	SRS 4.5	Team 5
10/26/2018	<5.0>	SRS 5.0	Team 5
11/09/2018	<5.5>	SRS 5.5	Team 5
11/19/2018	<6.0>	SRS 6.0	Team 5
11/23/2018	<7.0>	SRS 7.0: Release Version	Team 5

Table of contents

List of figures

1. Introduction

The introduction of the Software Requirements Specification document (SRS) provides a detailed overview of the entire document, as it covers the purpose, scope, definitions, acronyms, abbreviations, references, as well as an overview of the full document. This document serves as a means of assembling, analyzing and giving a thorough description of the entire **Biblio** software system, by going over the product features as well as the requirements set by the stakeholders. It also focuses on the needs of the stakeholders while presenting the constraints, requirements, and features of the product simultaneously.

Purpose

The purpose of this document is to build an online system to manage a library catalog that includes printed materials such as books, as well as unprinted materials such as movies. It is restricted within the premises of the Concordia University and serves as a means of facilitating the management of a library catalog. This document lists all the functional and non-functional requirements of this web application and provides a detailed overview of the constraints and tools used throughout the project. This project has been implemented under the guidance of Dr. C. Constantinides, and is intended for use by librarians (administrators) as well as users subscribed to the library.

Scope

The main objective of this web application is to facilitate the management of library cataloguing and to build a useful and simple application for library workers and users. Such as, registering materials into the system, or performing tasks: loaning/returning materials to the library. The system is based on a relational database as it facilitates catalog management and allows clients to loan and returned borrowed materials.

Definitions, acronyms, and abbreviations

- SRS: Software Requirements Specifications Document.
- UML: Unified Modeling Language.
- ER Model: Entity-Relationship Model
- CRUD: Create, read, update or delete.
- SSD: System Sequence Diagram.
- EFSM: Extended Finite State Machine.
- EJS: Embedded JavaScript Templates.
- Node.js: Runtime based on JavaScript.
- Express.js: Web application framework for node.js
- Heroku: Cloud platform used to build and monitor web applications.
- PostgreSQL: Open source object-relational database system.
- ISBN: the International Standard Book Number.
- ASIN: Amazon Standard Identification Number.
- AWS: Amazon Web Services.

2. Overall description

This section describes a background to the requirements: The general factors that affect the product, such as constraints, assumptions and dependencies.

Product Perspective

The product has a client-server architecture, and it is considered to be self-contained since it does not rely on any other software to function fully. The relational database system stores the following information:

- **Client record:**

  A client record consists of a unique id, a first name, a last name, a physical address, an email address, as well as a phone number, a discriminator for whether it is a client or an administrator, and lastly their password.

- **Book record:**

  A record for a book contains a title, an author, format (paperback or hardcover), a total number of pages, a publisher, a year of publication, a language and two ISBNs (a unique identification code) 10- and 13-digit, respectively.

- **Magazine record:**

  A record for a magazine contains a title, a publisher, a date of publication, a language and an ISBN.

- **Movie record:**

  A record for a movie item consists of a title, a director, a producer, actor's information, a language, subtitles, a dubbed (can be null if not dubbed), release date and run-time.

- **Music record:**

  A record for a music item consists of a type, a release date, a title, an artist, a label and ASIN.

Product functions

The **Biblio** system is a library that maintains a catalog of multitude of items, both printed and unprinted. It allows the clients to view the content of the catalog as well as borrow and return items, depending on their availability. Furthermore, administrators can use the system to enter view items, modify or delete existing records as well as create new items. In other words, the administrator can execute all CRUD operations. Also, administrators can also perform library searches, manage the all users of the system as well as view a list of the active ones. However, they are not allowed to loan or return items. As for the system itself, it keeps records transactions, such as loans and returns. Records

contain user's information as well as the transaction date. These transaction records cannot be overridden by the administrators.

User characteristics

The intended users of the system are individuals that are: members of the library with a valid user ID and password that are recognized by the library's database. It is assumed that users have access to a computer, which is connected to the internet, have already used and navigated on an computer, and can log into their respective accounts. Additionally, users must be aware that their loans are due on a specific date. If they fail to return a borrowed item on time, penalties may be applied. Finally, users are not expected to have any previous experience with operating the web application, as it is user-friendly, and does not require any special training to be used.

Constraints

There are many constraints that limit the options that the developers have. They can be summarized as follows:

- **Time**: it is very critical to for the developers to finish all their assigned tasks on time, given that the project is divided into six sprints.
- **Copyright issues:** Developers need to make sure that they are not infringing on any copyrighted materials when it comes to the items present in the library catalog.
- **Safety and security**: developers need to make sure that the software is secure.
- **Budget**: developers need to make sure to remain within the budget limits. In this project, the database as well as the server used are free.

Assumptions and dependencies

It is assumed that all users of the software system have a functional computer, an internet connection, as well as a modern browser. No specific operating system is required to be able to use the web application, although its functionality could be limited on an outdated browser such as *Internet Explorer*.

3. Specific Requirements

3.1. External Interfaces

3.1.1. User Interfaces

The navigation bar, shown below in Figure 1, is present at all times fixed to the top of the page. Depending on if a user or an administrator is logged in it will display different menus. Once a user or administrator is logged the Register and Login buttons are no longer be visible.

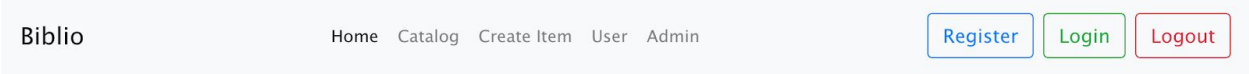

Figure 1: Navigation Bar with no user logged in.

Figure 1 below shows a view of the Catalog page. This page shows all the items in the catalog, as well as the option to search or filter the catalog.

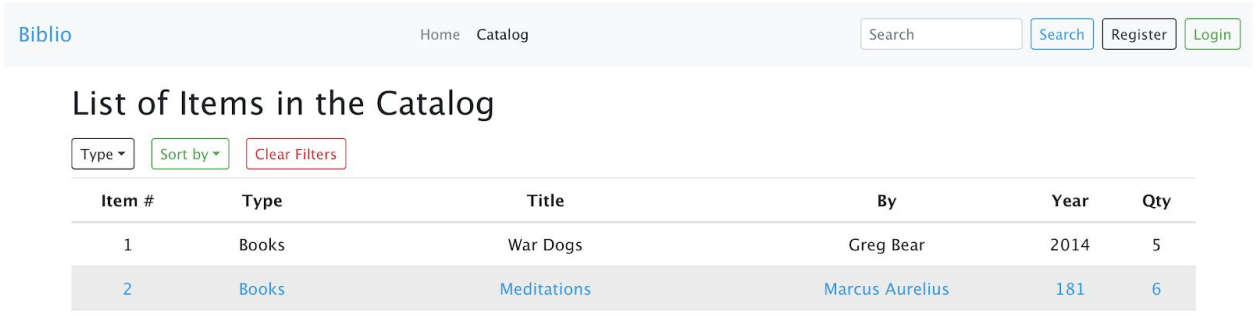

Figure 2 - A view of Biblio's Catalog page

Figure 2 below shows an administrators' page to promote or demote users.

Figure 3: Admin page to promote/demote users administrator privileges

The login page below shown in Figure 4 shows an auto generated error.

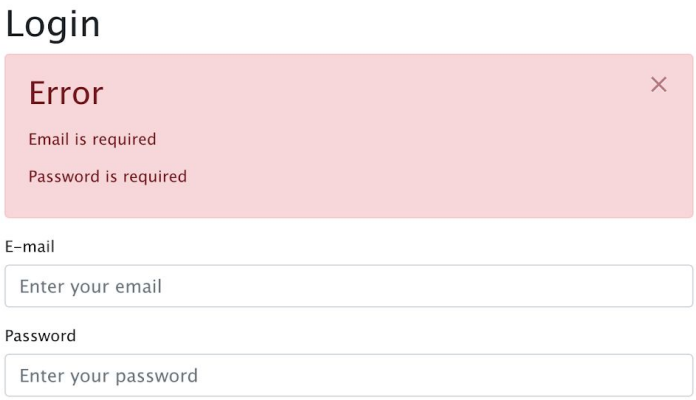

Figure 4: Login page displaying an error using the error partial

3.1.2. Software Interfaces

This section will list the software interfaces **Biblio** is using; the table below will list versions and websites. **Biblio** is a webapp using a Node.js runtime with Express.js as a framework. The webapp is hosted on a free Heroku server labeled as a Free Dyno. A Free Dyno is a server that goes to sleep after 30 minutes of inactivity, it will then need to be woken up for a request after those 30 minutes. The server is running on a Linux platform, specifically the Ubuntu operating system. It makes use of a PostgreSQL database management system hosted on AWS (Amazon Web Services) through Heroku. The PostgreSQL database is offered on Heroku as a software as a service, allows for scalability, and offers up to 10 000 rows in the free tier. The views discussed in section 3.1.1 are rendered using EJS as a templating engine together with Bootstrap as a front-end CSS framework.

Software	**Version**	**Website**
Node.js	v-8.12.0	https://nodejs.org/
Express.js	v-4.16.4	https://expressjs.com
Heroku Free Dyno	v-heroku-18	https://devcenter.heroku.com/articles/dyno-types
Ubuntu	v-18.04	https://www.ubuntu.com
PostgreSQL	v-10.5	https://www.postgresql.org
EJS	v-2.6.1	http://www.ejs.co
Bootstrap	v-4.1.3	http://getbootstrap.com

Figure 5: Table showing the different software interfaces of the system, their versions, and websites.

3.1.3. Communications Interfaces

Communication between the views and the server consist of creating, reading, updating, and deleting operations to and from the database.

3.2. Functional Requirements

Actor	Goal
User	- Ability to register as a new registered user. - Ability to filter catalog. - Ability to search through the catalog and apply a filter to the search.
Registered User	- Ability to view the inventory. - Ability to loan an item. - Ability to return an item.
Administrator (a type of user)	- Ability to enter item(s) into the Catalog. - Ability to modify item(s) in the Catalog. - Ability to delete item(s) from the Catalog. - Ability to register other administrators. - Ability to promote/demote other administrators. - Ability to view all users in the system. - Ability to view active users.

Figure 6: Table showing the functional requirements of the system.

3.3. Use Case Models

3.3.1. CRUD Use Case Model

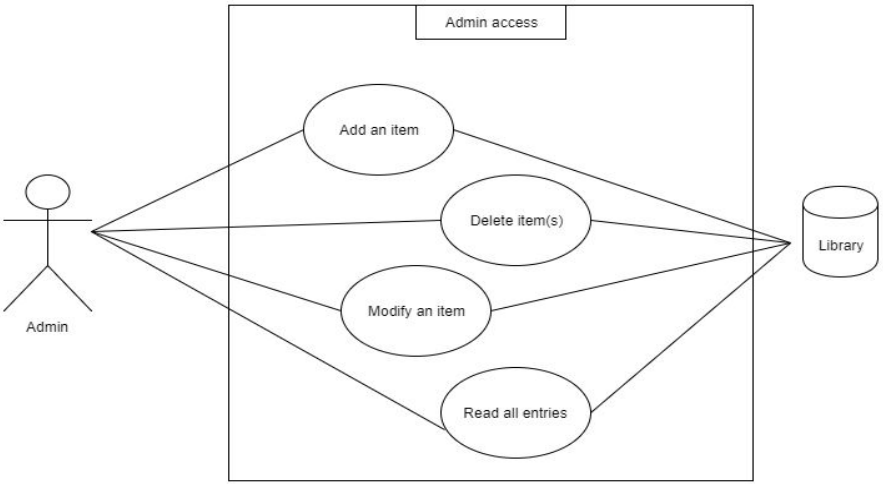

Figure 7: CRUD Use Case Model

Primary Actor: System Administrator

Stakeholders and Interests: System administrator:

- Wants to create an entry to add to the library.
- Wants to read added entries.
- Wants to update (modify) added entries.
- Wants to delete added entries.

Preconditions:

- User is logged is authenticated as an administrator for the following actions:
- **Create**:
    - Entry doesn't already exist.
- **Read**:
    - The object has to exist in the database.
- **Update**:
    - Modified values have to be valid.
- **Delete**:
    - Entry must exist before deleting.

Success Guarantee (Postconditions): The newly created entry is read, modified and is deleted from database afterwards.

Main Success Scenario:

1. Administrator adds a valid entry.
2. Administrator views the newly created entry.
3. Administrator makes a change to the ISBN of the entry.
4. Administrator deletes the newly modified entry.

Extensions:

1. If the entry already exists, the system throws an error.
2. If a modified value is invalid, the system throws an error depending on the input field.
3. If the entry to be modified/deleted/read doesn't exist, the system throws an error.

3.3.2. Search and Filter Use Case Model

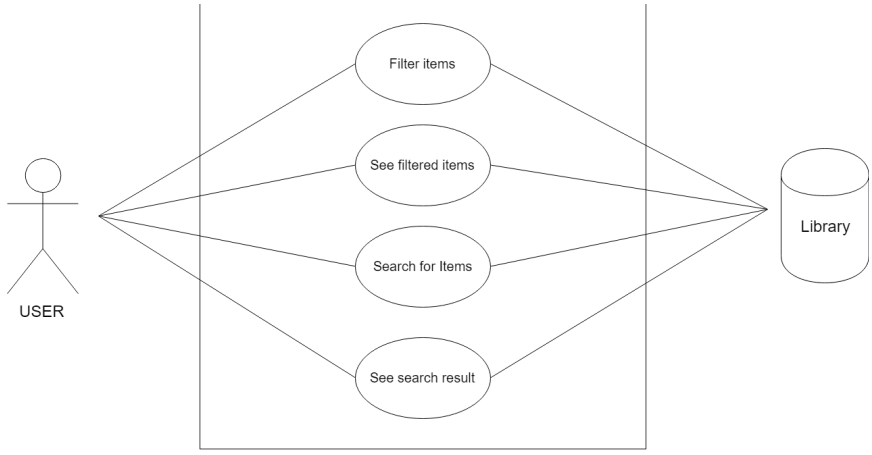

Figure 8: Search and Filter Use Case Model

Primary actor: user

Stakeholders and Interests: user, system administrator:

- Wants to search for (a) specific item(s).
- Wants to view search results.
- Wants to view items following a sorting criteria by applying specific filters.
- Wants to apply the filters on the viewed search results.

Preconditions:

None.

Success guarantee (postconditions):

Search: A list of items containing the keywords used by the user is displayed.

Filter: A list of filtered items that follow the selected sorting criteria.

Main success scenario:

1. User filters items by selecting the sorting criteria.
2. A list of filtered items is displayed.
3. User searches for an item.
4. A list of items containing the keywords used by the user is displayed.
5. User selects desired filtering criteria on the searched items.
6. A filtered list of items containing the keywords entered by the user is displayed following the selected filters.

Extensions:

1. If the item does not exist, the system shows no results found.

3.3.3. Loan Use Case Model

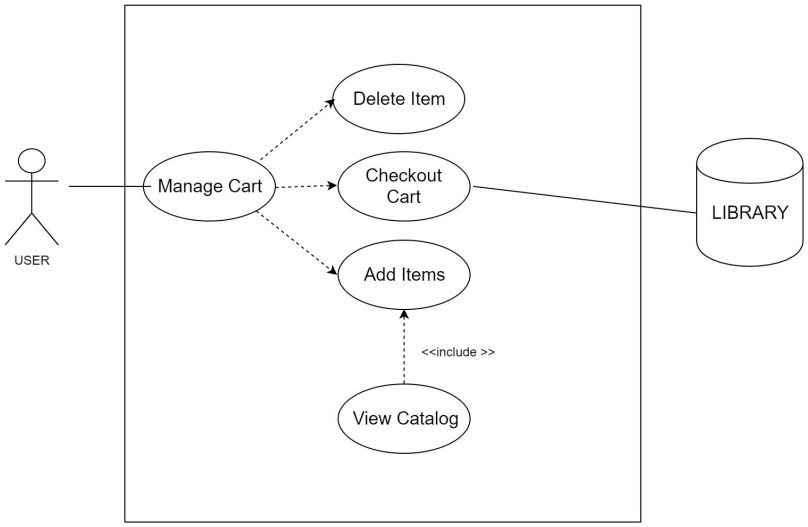

Figure 9: Loan Use Case Model

Primary Actor: registered user.

Stakeholders and Interests: registered user:

- Wants to borrow (an) item(s) from the library.

Preconditions:

- User is logged and authenticated.
- Item(s) to be borrowed are available.

Success guarantee (postconditions): User borrows desired item(s) successfully.

Main success scenario: User borrows desired item(s).

Extensions:

1. If the item does not exist, the system throws an error.
2. if the item is not available, the system throws an error.
3. If the number of items to be borrowed exceeds the maximum number of items allowed, the system throws an error.

3.3.4. Temporal Logic for Loan and Return Use Case

Loan Item Temporal Logic

$\square$ [(IS_USER $\wedge$ FULL_CART) $\Rightarrow$ CHECKOUT_CART]

Return Item Temporal Logic

$\square$ [(ITEM_LOANED $\wedge$ IS_RETURNED) $\Rightarrow$ RETURN_ITEM]

3.4. Non-functional requirements

3.4.1. Performance efficiency

The system does not require high hardware specifications. The system should be able to respond to user requests in less than 5 seconds.

3.4.2. Compatibility

The system is fully compatible with the following web browsers : Google Chrome, Firefox and Safari.

3.4.3. Usability

Users are required to know how to use a computer and have a basic knowledge on how to browse a website. Users should be able to use the software and do the basic operations described in the use cases. Users are provided with the documentation to help if needed.

3.4.4. Reliability

The system is able to do the following operations signup and login a user, CRUD, search for an item and loan an item.

3.4.5. Security

The system must resist unauthorized usage, and grant access to only registered users. Passwords are hashed in the database.

3.4.6. Maintainability

The software uses Node.js which is a runtime based on JavaScript therefore the application is highly reusable and will function in different environments. The system structure can be easily adapted to different requirements
Also the software application can be easily modified without any defects because of the effectiveness and efficiency of the code.

3.4.7. Portability

The software can run on multiple environments with different operating system including windows, linux and MacOS.

3.4.8. Design constraints

Decisions that must be followed, such as languages, processes, prescribed use of tools, architectural and design constraints, purchased components, class libraries, etc.

4. Analysis Models

4.1 System Sequence Diagram (SSD) for CRUD Use Case

1. Create a New Item

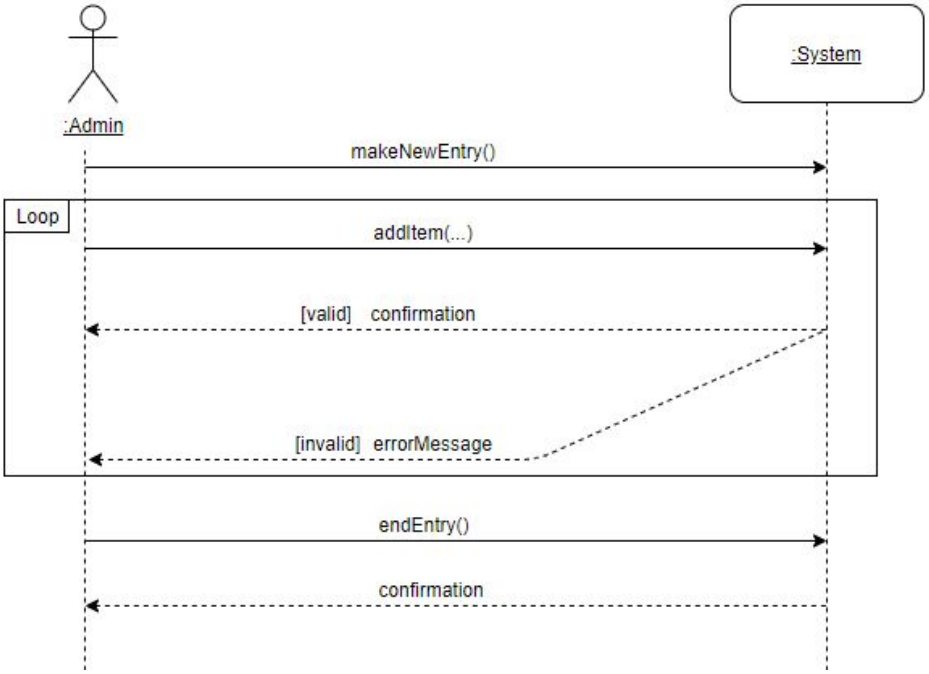

Figure 10: CreateItem Sequence Diagram

2. Read an Item

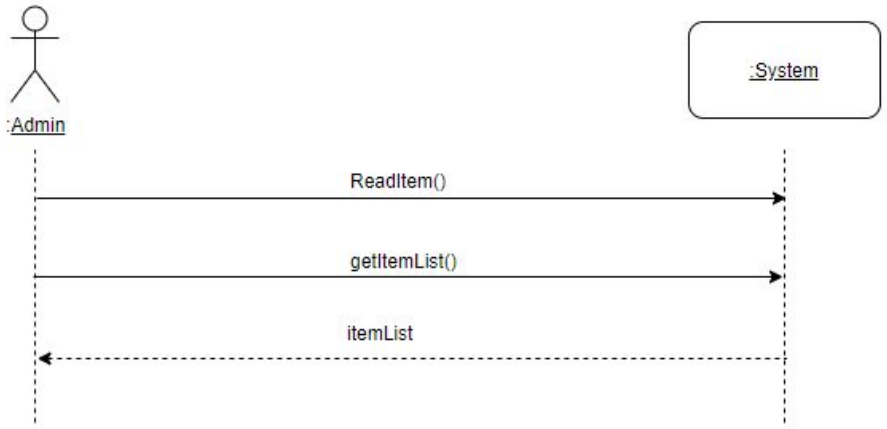

Figure 11: Read Item Sequence Diagram

3. Modify an Existing Item

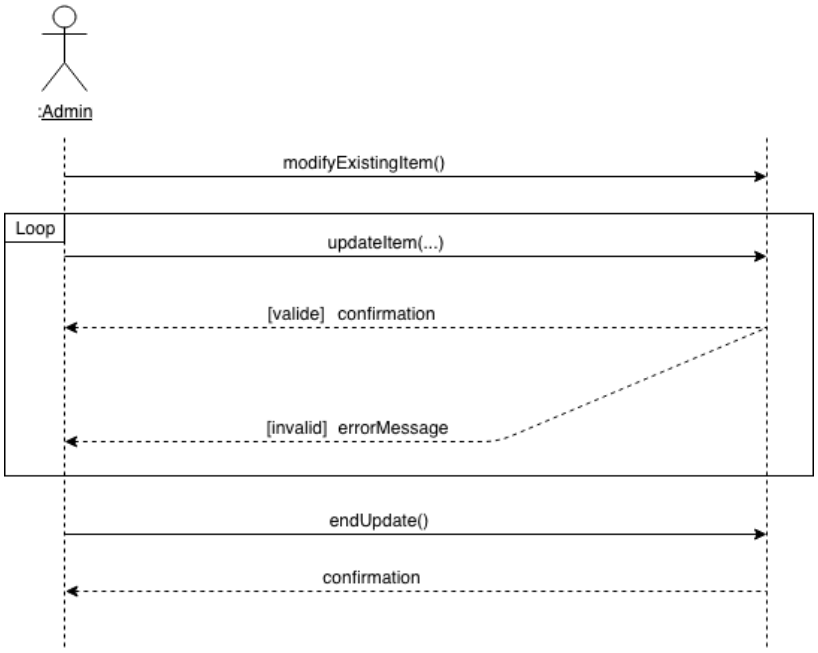

Figure 12: Update Item Sequence Diagram

4. Delete an Existing Item

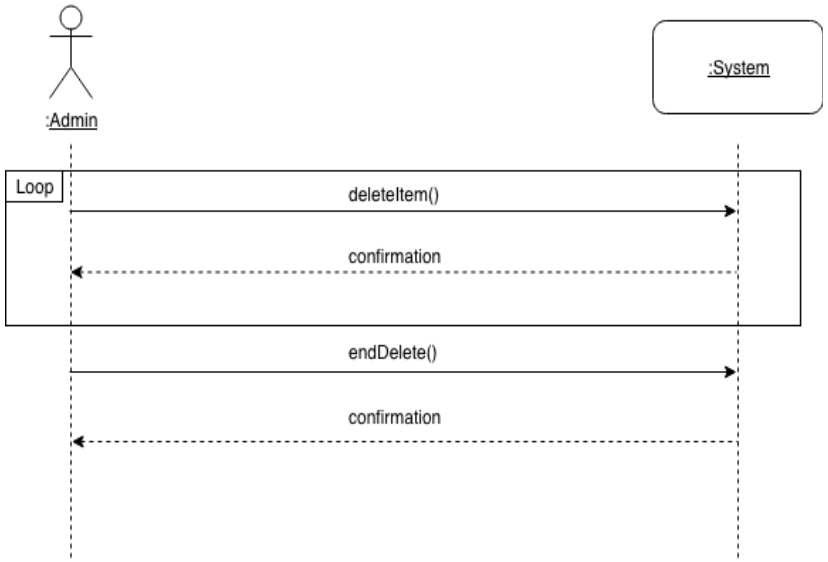

Figure 13: Delete Item Sequence Diagram

5. Search The Catalog for items

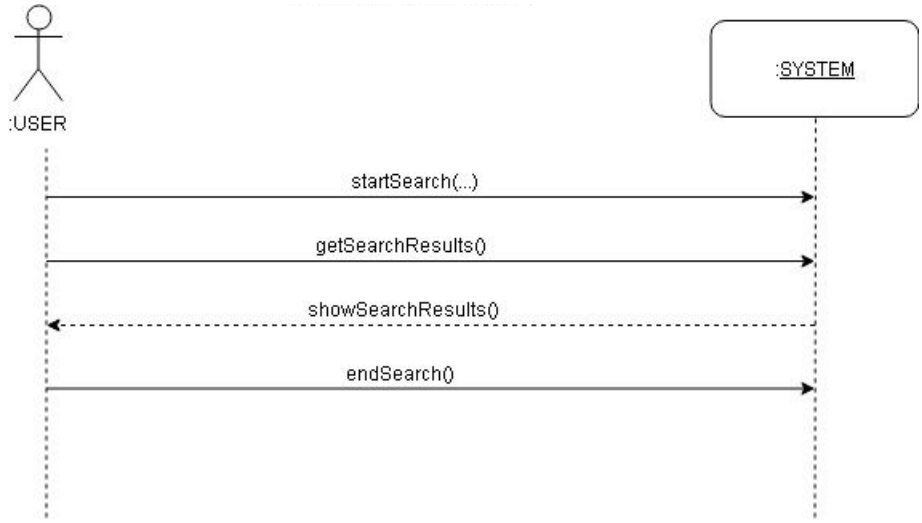

Figure 14: Search Items Sequence Diagram

6. Filter Catalog Items

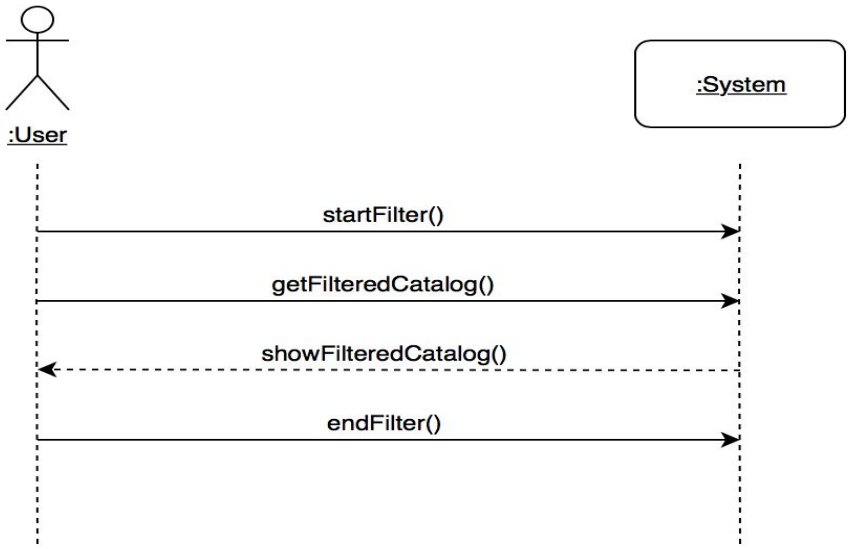

Figure 15: Filter Items Sequence Diagram

7. Loan an item

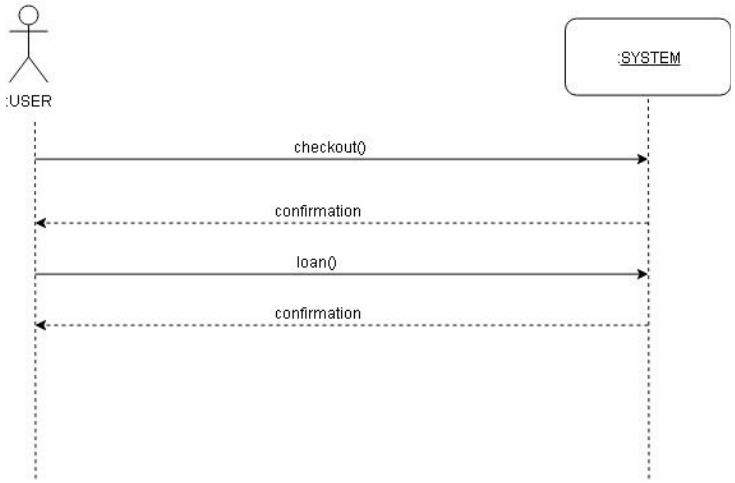

Figure 16: Loan Item SSD

4.2 Contracts

Contract CO2:	**createItem**
Operation:	createItem (itemAttributes: Array)
Cross Reference:	Use Case: CRUD
Pre-conditions:	User is admin
Post-conditions:	1. An item instance i was created 2. i was associated with the Catalog

Figure 17: createItem Contract

Contract CO2:	**readItemList**
Operation:	readItemList()
Cross Reference:	Use Case: CRUD
Pre-conditions:	User is admin. Catalog already exists.
Post-conditions:	1. View was associated to Catalog

Figure 18: readItemList Contract

Contract CO3:	updateItem
Operation:	updateItem(id)
Cross reference:	Use case: CRUD
Preconditions:	1. User is identified and authenticated as admin. 2. Item with id exists.
Postconditions:	1. Item i is associated with Catalog. 2. Item i with identifier id has a new value.

Figure 19: updateItem Contract

Contract CO4:	deleteItem
Operation:	deleteItem(id: itemID)
Cross reference:	Use case: CRUD
Preconditions:	1. User is identified and authenticated as admin. 2. Item **a** to be deleted exists.
Postconditions:	1. Association between item **a** andCatalog is broken. 2. Change is saved: item a is deleted.

Figure 20: deleteItem Contract

Contract CO5:	searchItem
Operation:	searchItem(String: searchQuery)
Cross reference:	Use case: Search
Preconditions:	None.
Postconditions:	View is associated with Catalog items similar to query.

Figure 21: searchItem contract

Contract CO7:	**filterItems**
Operation:	filterCatalog (String: filterType)
Cross Reference:	Use Case: Search and filter
Pre-conditions:	None.
Post-conditions:	View is associated with Catalog items sorted according to the selected sorting criteria.

Figure 22: filterItem Contract

Contract CO7:	loanItem
Operation:	loanItem(id)
Cross reference:	Use case: Loan
Preconditions:	1. User is logged in and authenticated (not an admin)
	2. Cart is not empty.
Postconditions:	1. Instance t of Transaction is created.
	2. t is associated with Transactions.
	3. Cart is empty

Figure 23: LoanItem Contract

4.3 Partial Domain Model

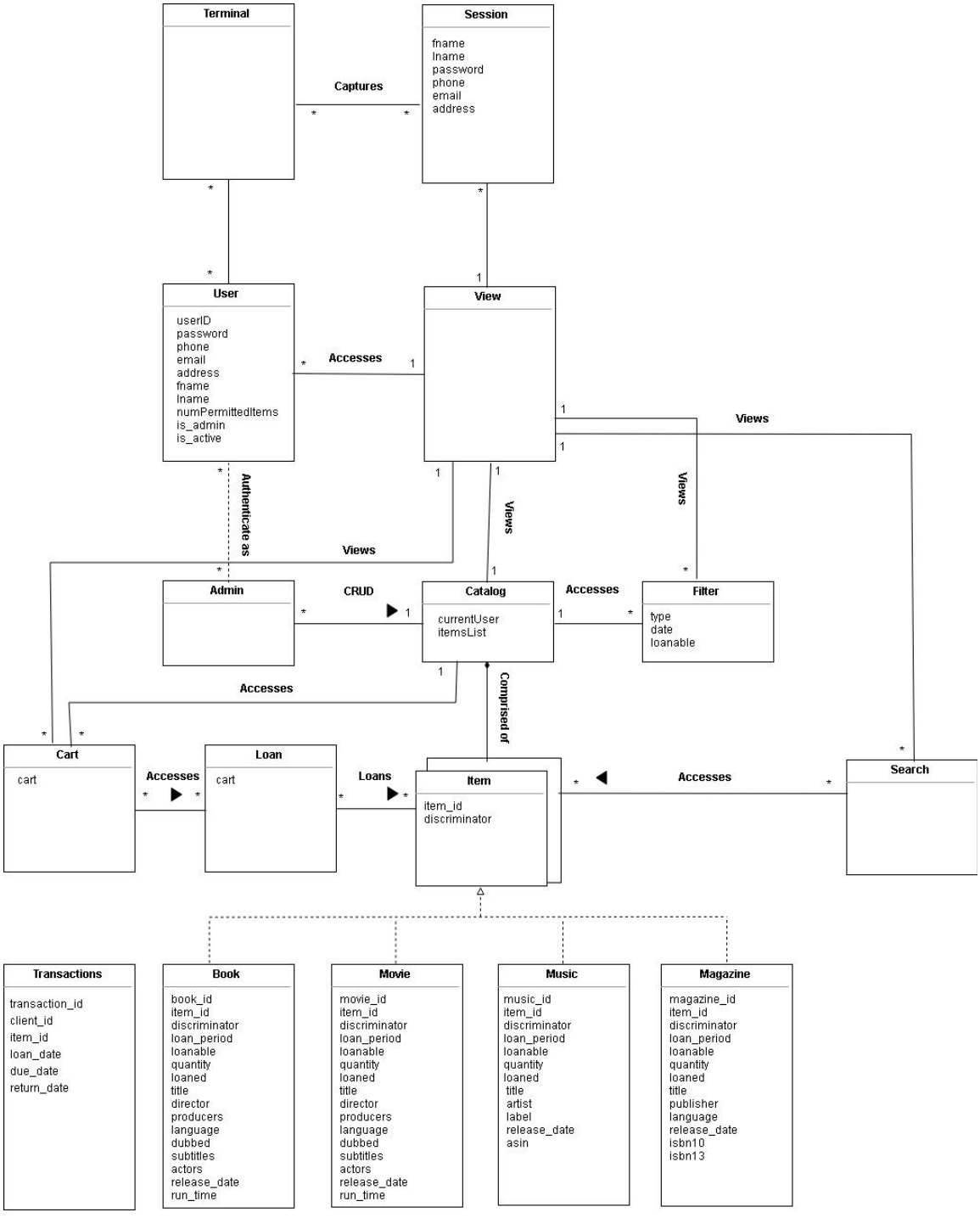

Figure 24: Domain Model

4.3 Extended Finite State Machine for Loan Use Case

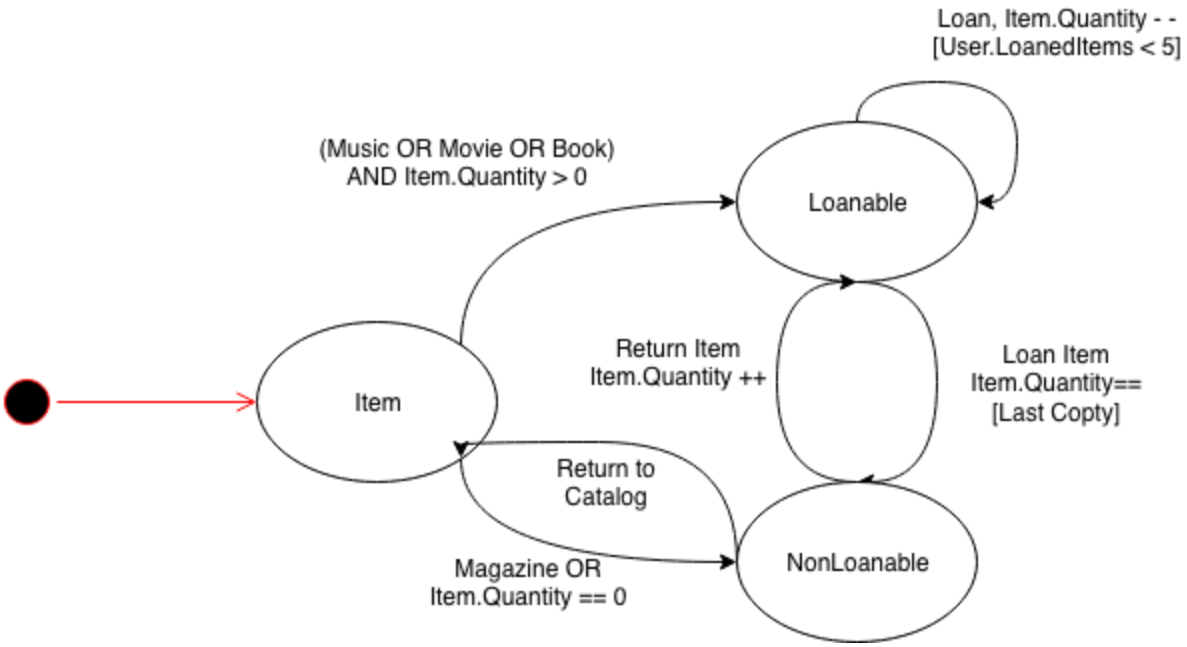

Figure 25: LoanItem Use Case EFSM