http://www.cs.concordia.ca/~ormandj/comp354/2003/Project/ieee­SDD.pdf

(Team Name)

**Online Library**

Software Design Document

Name (s):  Ciacâru Cristian-Dumitru

Scăunașu Claudia-Nicoleta

Tănăsuică Alexandru-Constantin

Lab Section:  -

Workstation: -

Date: (03/19/2021)

**TABLE OF CONTENTS**

**1. INTRODUCTION  2**

1.1  Purpose

1.2  Scope

1.3  Overview

  1.4  Reference Material

1.5  Definitions and Acronyms

**2. SYSTEM OVERVIEW  2**

**3. SYSTEM ARCHITECTURE 2**

3.1  Architectural Design

  3.2  Decomposition Description

  3.3  Design Rationale

**4. DATA DESIGN**

4.1  Data Description

  4.2  Data Dictionary

**5. COMPONENT DESIGN  3**

**6. HUMAN INTERFACE DESIGN**

6.1  Overview of User Interface

6.2  Screen Images

  6.3  Screen Objects and Actions

**7. REQUIREMENTS MATRIX  4  8. APPENDICES  4**

**8. Appendices**

**1. INTRODUCTION**

**1.1  Purpose**

This Software Design Document describes the architecture and system design of the Online Library application. It is a translation of requirements into a description of the software structure, software components, interfaces and data necessary for the implementation phase. In essence, the SDD becomes a detailed blueprint for the implementation activity done by the designers.

It is anticipated that the SDD will be used by the application designers. Designers will use the information recorded here as the basis for creating the structure and design of each component.

**1.2  Scope**

The scope of this product is to bring the classical library to everyone without the necessity of going to a specific place to see the available books in order to borrow them. Within this application, we want to make a friendly environment for the readers, where they can easily access any of the application features and make their “stay” as comfortable as we can.

The application is intended to provide the following goals and objectives:

→ To build a system that can receive input and automatically generate output in an easy way and short time.

→ Simplify search/discovery of library resources.

→ To build a monitoring system that is able to monitor and manage all library operations efficiently.

→ To enter and preserve details of the various issues and keep a track on their returns

→ The ability to manage the book inventory database including remove, change, and add through PC.

→ The application will provide to the users the search function on books based on subject, title, or author.

→ Provide the possibility of registration online at any desired time

The 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.

→ The availability of information at any time in any place

**1.3  Overview**

This document completely describes the system at the architecture level, including subsystems and their services, data management, component design which will describe what each component does in a more systematic way and human interface design which will provide the functionality of the system from the user’s perspective . The document is organized into eight major sections. Each section provides detailed sub-sections relevant to the major section. Charts, tables, and graphics have been inserted to explain or clarify content

**1.4  Reference Material**

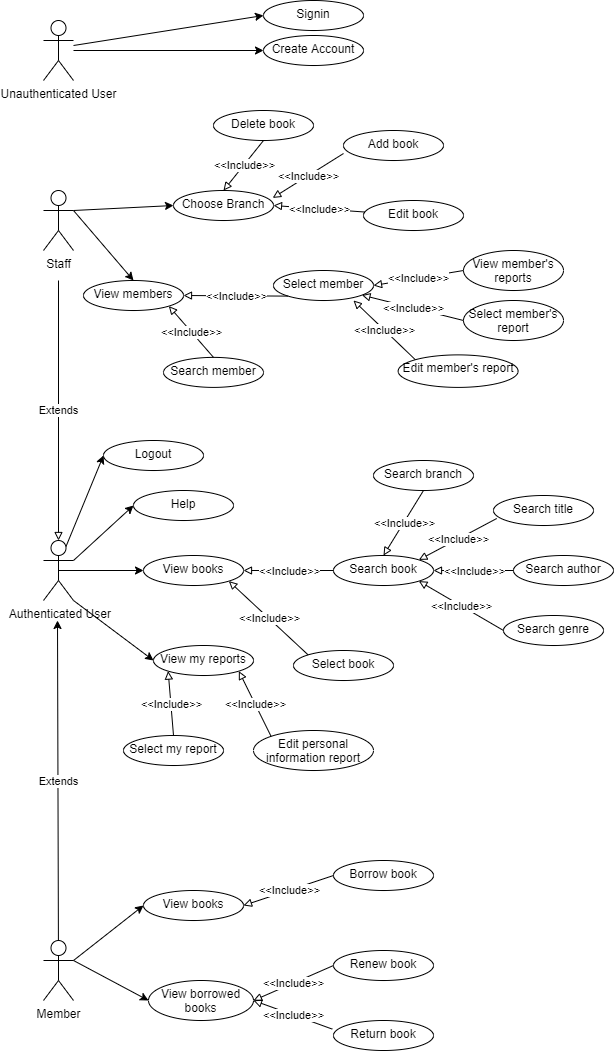
*This section is optional.*

List any documents, if any, which were used as sources of information for the test plan.

**1.5  Definitions and Acronyms**

|  |  |  |
| --- | --- | --- |
| **No. Crt.** | **Abbreviation** | **Meaning** |
| 1 | SDD | Software Design Document |
| 2 | WPF | Windows Presentation Foundation |
| 3 | XAML | Extensible Application Markup Language |
| 4 | MVVM | Model View ViewModel |
| 5 | UI | User Interface |
| 6 | UX | User Experience |
| 7 | GUI | Graphical User Interface |
| 8 | WCF | Windows Communication Foundation |

**2. SYSTEM OVERVIEW**



The Use Case Diagram will provide a basic overview of the online library application. There will be two main categories of users: authenticated and unauthenticated users. The number of actions of an unauthenticated user is limited since that person can only register or login.

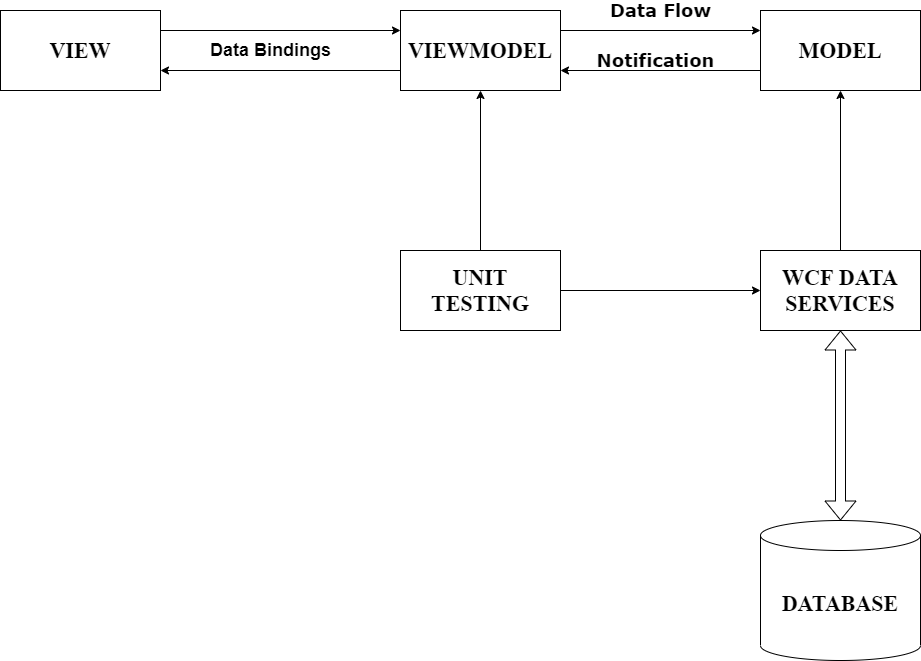
The authenticated users can also be of two types: members who are the readers registered in the library or staff who are people that are working in the library. There are some common functionalities for these two types of users. Both of them can see all the books, access information about a specific book and also search for a book.Branches can have different books, which is why it is important for users to have access to all the branches. The search has different filters such as: title, author,genre, branch. More than that, an authenticated user can access a help section in which there will be some information regarding how the application works. They can also view their own reports. There will be two types of reports: personal information report in which all the personal information of the user will be kept and history of borrowed books report where a person can see all the books he/she has issued since they are registered to this library. Of course, any user can edit the personal information report.

There are also some differences between the two types of authenticated users. Members can also borrow books from any branch they want. More than that, they will have a special section in which they can see all the books they have in possession and announce the intention of returning or renewing one book. On the other hand, staff members can add, delete or update books in the branches. They can also see the members and their reports. They have the authority to modify all types of reports of the members.

After finishing what they wanted to do, users can log out.

**3. SYSTEM ARCHITECTURE**

**3.1  Architectural Design**



**VIEW:** A View is defined in XAML and should not have any logic in the code-behind. It binds to the view-model by only using data binding.

**MODEL:** AModel is responsible for exposing data in a way that is easily consumable by WPF.

Windows Presentation Foundation (WPF) is a UI framework that creates desktop client applications. The WPF development platform supports a broad set of application development features, including an application model, resources, controls, graphics, layout, data binding.

**VIEWMODEL:** A ViewModel is a model for a view in the application or we can say as an abstraction of the view. It exposes data relevant to the view and exposes the behaviors for the views, usually with Commands.

The View is bound to the properties of the ViewModel, which in turn exposes the data contained in the model object and other states specified in the view. The binding between View and ViewModel is very simple to construct because a ViewModel object is set as the DataContext of a view. If the value of the property in the ViewModel changes, those new values ​​are automatically propagated to the view via data binding.When the user clicks on a button on the View, a command on the ViewModel performs a requested behavior. ViewModel, not View, performs all modifications to the model data.

The View class doesn't know the existence of the model class, and neither the ViewModel nor the Model knows the view. In fact, Model completely forgot the fact that ViewModel and View exist. This is a very loosely coupled design.

Windows Communication Foundation (WCF) is a framework for building service-oriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another. A service endpoint can be part of a continuously available service hosted by IIS, or it can be a service hosted in an application. An endpoint can be a client of a service that requests data from a service endpoint. The messages can be as simple as a single character or word sent as XML, or as complex as a stream of binary data.

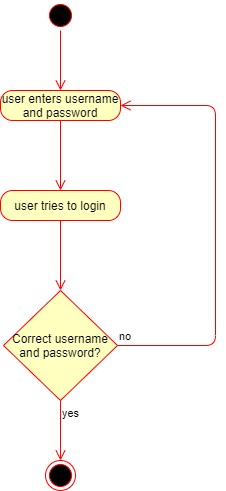
**3.2  Decomposition Description**

The Model View ViewModel (MVVM) is an architectural pattern used in software engineering.

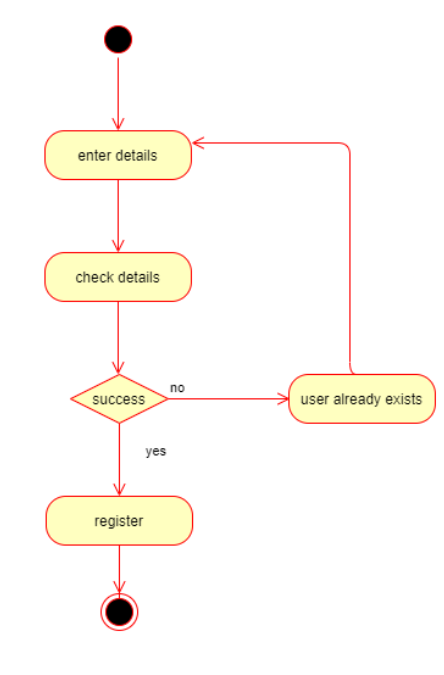
It is a way of creating client applications that leverages core features of the WPF platform and allows for simple unit testing of application functionality.

* VIEW: A view is defined in XAML and should not have any logic in the code-behind.It binds to the view-model by only using data binding.
* MODEL: A Model is responsible for exposing data in a way that is easily consumable by WPF.
* VIEWMODEL: A ViewModel is a model for a view in the application or as an abstraction of the view. It exposes data relevant to the view and exposes the behavior for the views, usually with Commands.

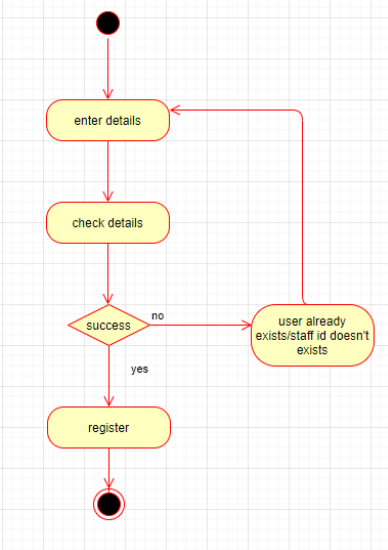
**1.User Login** - both member and staff should introduce the username and password and hit the login button. If everything is ok they will be redirected to the homepage, otherwise they will have to introduce the username and password again.



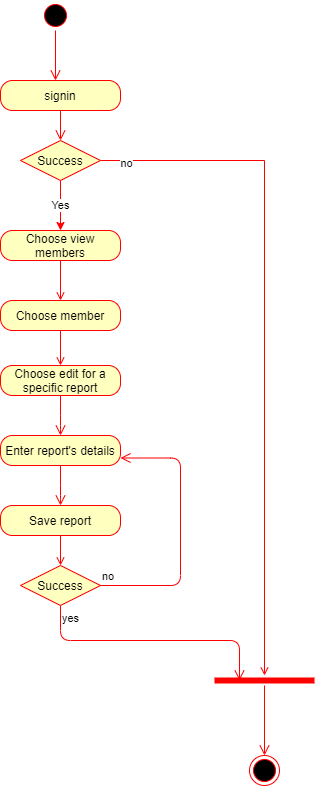
**2.User registration** - if the person is an unauthenticated user, for performing any action he/she must register. The new user must enter the details, then the details will be checked and if some details are already used by another user, then the person will be asked to introduce new data. If everything works alright the person is registered. The person will be redirected to the main page after authentication.



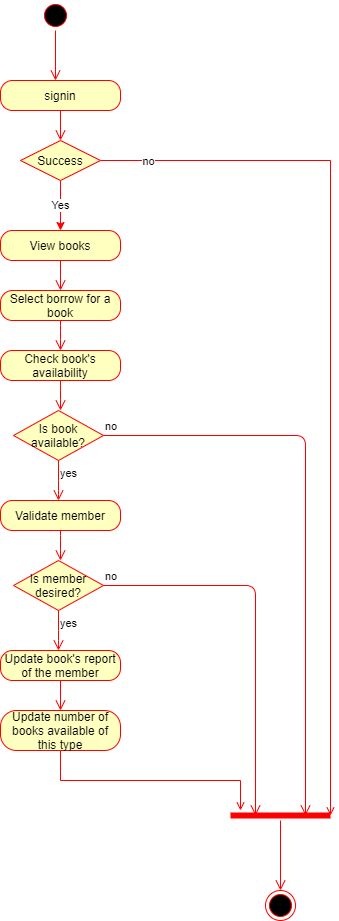
**3.User registration** - if the person is an unauthenticated user, for performing any action he/she must register. The new user must enter the details, then the details will be checked and if some details are already used by another user, then the person will be asked to introduce new data. The same thing happens if the id required for the staff is wrong. If everything works alright the person is registered. The person will be redirected to the main page after authentication.



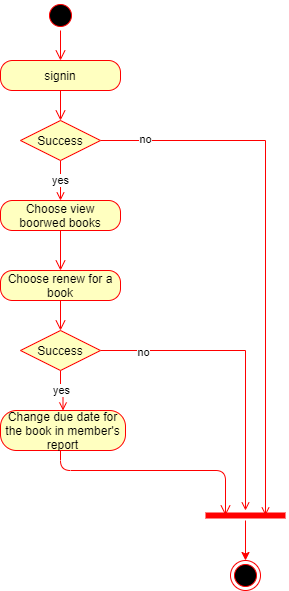
**4.Edit Member’s Report** - this action will be done by staff users. Only after they successfully login, they can choose to view the members, select one member in order to see his/her reports. After that, staff will see an “Edit” button besides each report. They can press on any “Edit” button they want, enter the new details of the report and save the modifications. If some fields will remain incomplete, when clicking the “Save” button they will be warned that they need to fulfil a specific field. If everything works alright, after saving the new information added, the staff will be redirected to the member’s updated report.



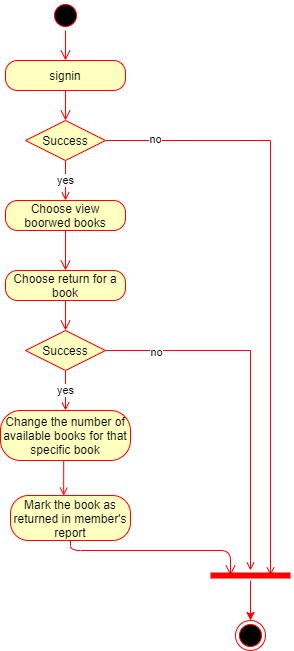
**5.Borrow book** - members can borrow books but only after they successfully logged in. After this step they have to choose a branch from where they want to borrow a book. After selecting the view books and all the books will be displayed, so the person can press the “Borrow” button which is text to the book. After that button is pressed, the application will check if the book is still available for borrowing, then it will check if the member is desired (which means he/she never turned in a book later than the due date). After these steps are completed and they are successful, the book will be reserved for that member and this will modify both the quantity of books available and the history of borrowed books report. The member will be redirected to the books’ report page.



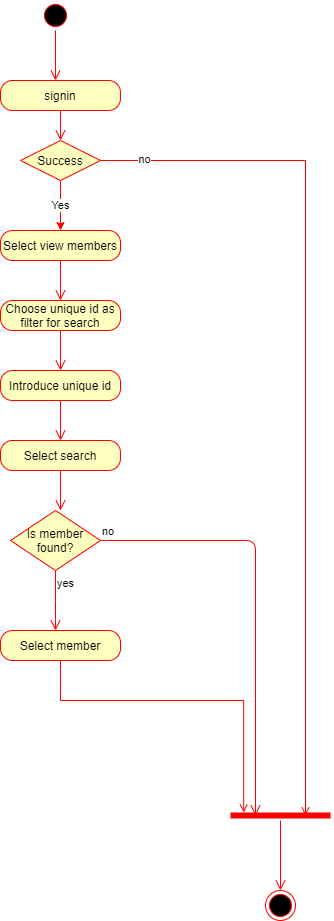
**6.Renew book** - the members have also the option of renewing a book they have in possession. After they successfully log in, they can choose to see the books they borrowed. Then, a list of borrowed books at this time will be available. The member can choose to prolong the due date by one week by clicking on the “Renew” button beside the book. If everything works fine, the due date in the borrowed books report will be changed. The member will be redirected to the report page.



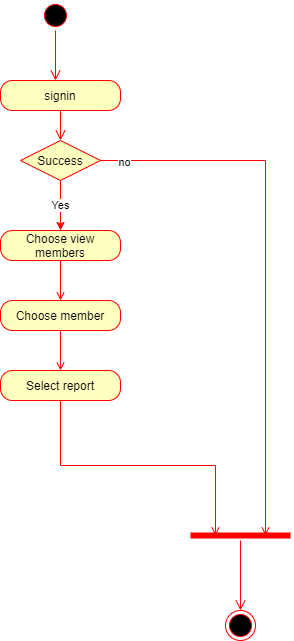
**7.Return book** - the members have also the option of returning a book they have in possession. After they successfully log in, they can choose to see the books they borrowed. Then, a list of borrowed books at this time will be available. The member can choose to return by clicking on the “Return” button beside the book. If everything works fine, the book will appear as returned in the borrowed books report and the number of available books of that type will be updated. The member will be redirected to the report page.



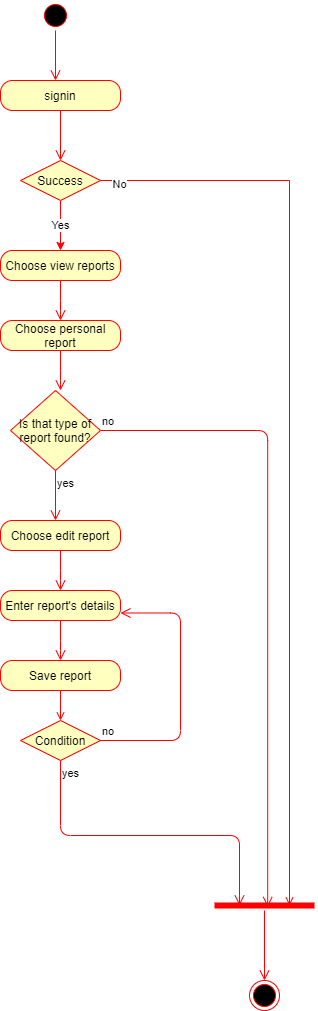
**8.Search member** - Staff users can search a member but only after they logged in. They will have to choose to view members and write in the search bar the unique id of the member. After that, they have to press the “Search” button. The application will search in the database for the member with that id and if it is found, the staff will see the member and can select it. The staff will be redirected to the member’s page.



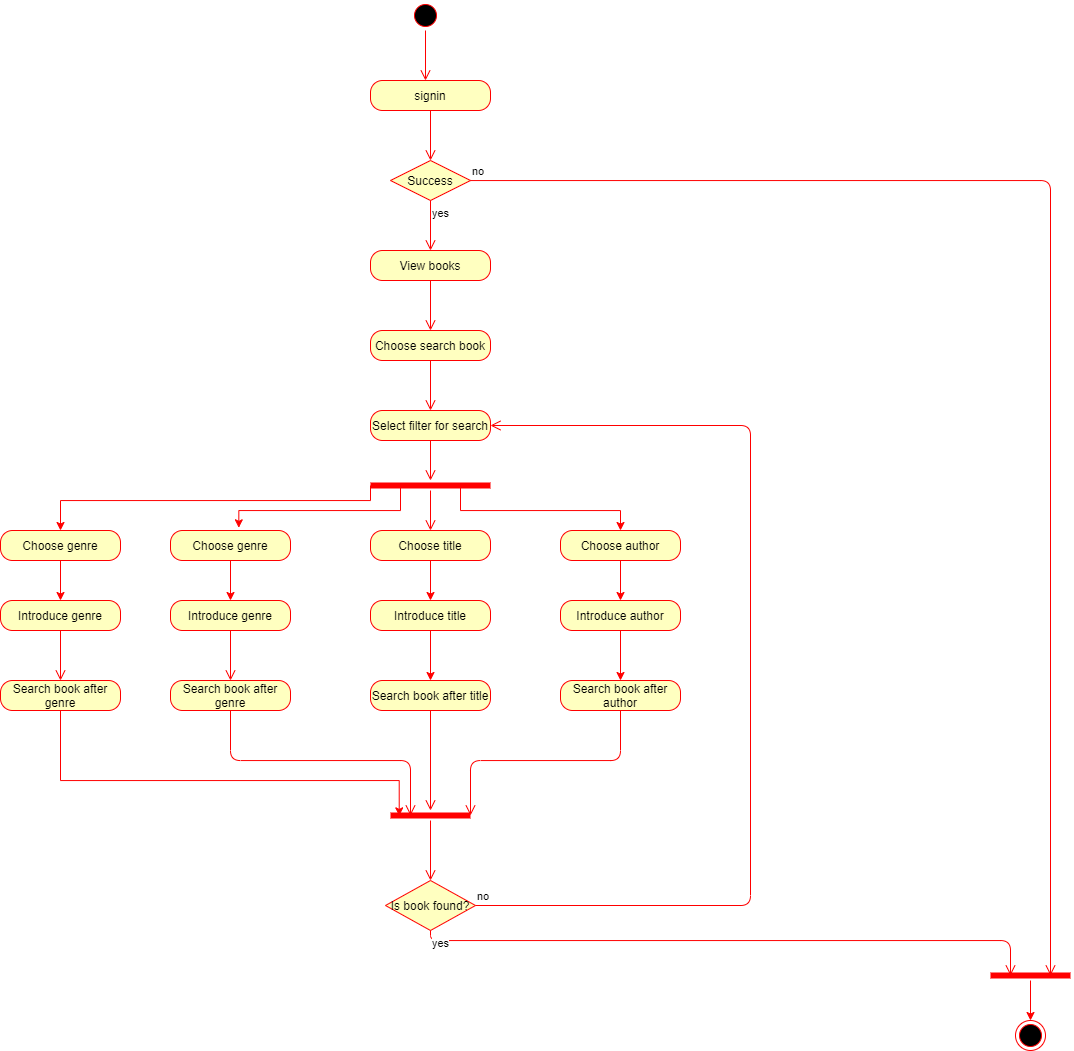
**9.Select member’s report** - Staff users can also select a member’s report. After they successfully log in, the staff should choose the view members option. From there a list with all members will be displayed. He/She can click on any member and from there he will go to the member’s report page. There the staff member can select any of the reports of the member. If everything works alright (which is in all cases) the staff user will be redirected to the member's report page.



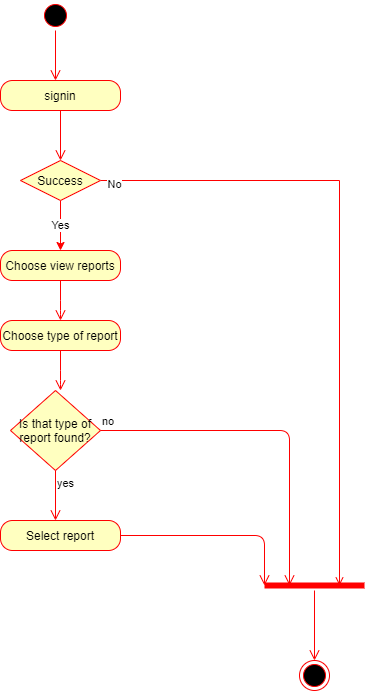
**10.User edit personal information report** - Both member and staff users can edit their personal information report, but only after they are successfully logged in. After this step, they will choose view reports and both types of users will have beside them an “Edit” button.They can enter the new details of the report and save the modifications. If some fields will remain incomplete, when clicking the “Save” button they will be warned that they need to fulfil a specific field. If everything works alright, after saving the new information added, the user will be redirected to the updated report.



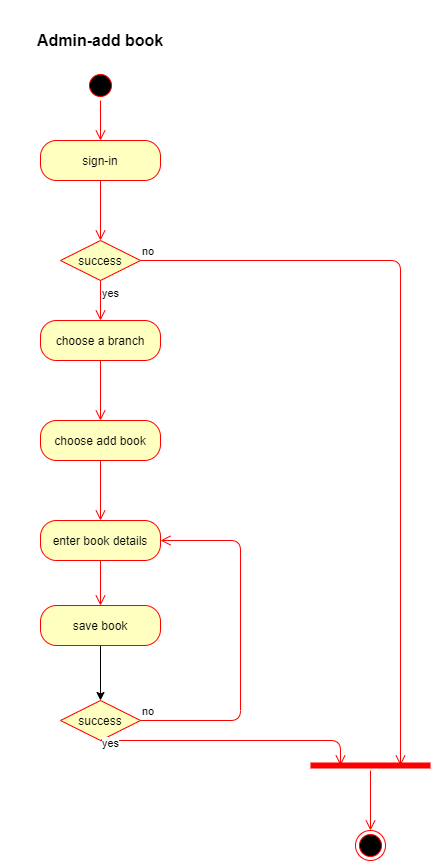
**11.User search -** Both member and staff users can search a book, but only after they are successfully logged in. After that step they have to select the filter. They have 3 possible filters: title, author, genre and branch . After selecting one of these filters, the user must introduce the information regarding the filter and select the “Search” button. If the book is found, then it will be displayed, otherwise the user will be asked to introduce again the data which will be searched.



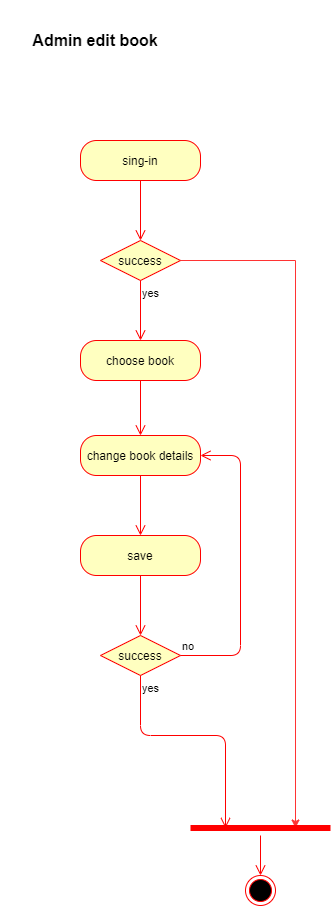
**12.User select report** - Both member and staff users can select a report, but only after they are successfully logged in. After this step, they can choose to view the reports, then from the list of reports, they can choose the type of report. If that report is found in the list, selecting the report will redirect the user to the chosen report’s page.



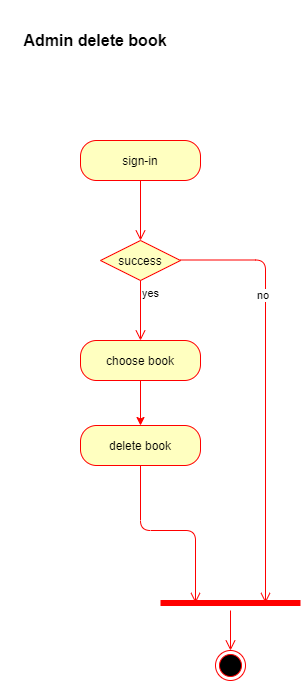
**13.Add book** - A staff member can add books to a branch. First of all, that person must log in. If the login is successful and choose to add a book. He/she must then complete the information about the new book and press the “Save” button. If some fields are not complete, the staff will be requested to complete them again, otherwise he/she will be redirected to the page with all books updated.



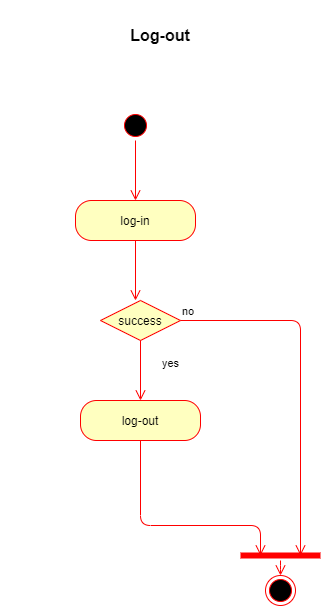
**14.Edit book** - A staff member can edit books. First of all, that person must log in. If the login is successful and choose to edit a book. He/she must then complete the new information about the book and press the “Save” button. If some fields are not complete, the staff will be requested to complete them again, otherwise he/she will be redirected to the page with all books updated.



**15.Delete book** - A staff member can delete books. First of all, that person must log in. If the login is successful. He/she must then press the “Delete” button besides the book wanted to be deleted. The staff will be redirected to the page with all books updated.



**16.Logout** - Both member and staff users can logout, but only after they are successfully logged in. All they have to do is press the “Logout” button.



**3.3  Design Rationale**

The MVVM architecture was chosen because this architecture is targeted at modern UI Development platforms (Windows Presentation Foundation) in which exist requirements that take into consideration the user experience (UX). The View-Model of MVVM is basically a “powerful value converter” meaning that the View-Model is responsible for exposing the data objects from the Model in such a way that those objects are easily managed and consumed. In this respect, the View-Model is more Model than View, and handles most if not all of the View’s display logic.

More than that, MVVM was designed to make use of specific functions in WPF to better facilitate the separation of View layer development from the rest of the pattern by removing virtually all “code-behind” from the View layer. Instead of requiring Interactive Designers to write View code, they can use the native WPF markup language XAML and create bindings to the ViewModel, which is written and maintained by application developers.

Furthermore, the MVVM architecture guides us how to distribute responsibilities between classes in a GUI application (or between layers), with the goal of having a small number of classes, while keeping the number of responsibilities per class small and well defined.

**4. DATA DESIGN**

**4.1  Data Description**

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed and organized. List any  databases or data storage items.

**4.2  Data Dictionary**

Alphabetically list the system entities or major data along with their types and descriptions. If you  provided  a  functional description  in  Section 3.2, list all the  functions and function  parameters. If you provided an OO description, list the objects and its attributes, methods and  method parameters.

**5. COMPONENT DESIGN**

In this section, we take a closer look at what each component does in a more systematic way. If

you gave a functional description in section 3.2, provide a summary of your algorithm for each  function listed in 3.2 in procedural description language (PDL) or pseudocode. If you gave an  OO description, summarize each object member function for all the objects listed in 3.2 in PDL or pseudocode. Describe any local data when necessary.

**6. HUMAN INTERFACE DESIGN**

**6.1  Overview of User Interface**

The application will have the following features:

1. Create account
2. Login user
3. Logout uses
4. User’s search functionality
5. User’s view report functionality
6. Staff user’s search member functionality
7. Staff user’s check member functionality
8. Staff user’s add book functionality
9. Staff user’s update book functionality
10. Staff user’s delete book functionality
11. User’s check book’s availability functionality
12. Member user’s renew book functionality
13. Member user’s return book functionality
14. User’s edit information functionality
15. User’s choose branch functionality

Used frameworks:

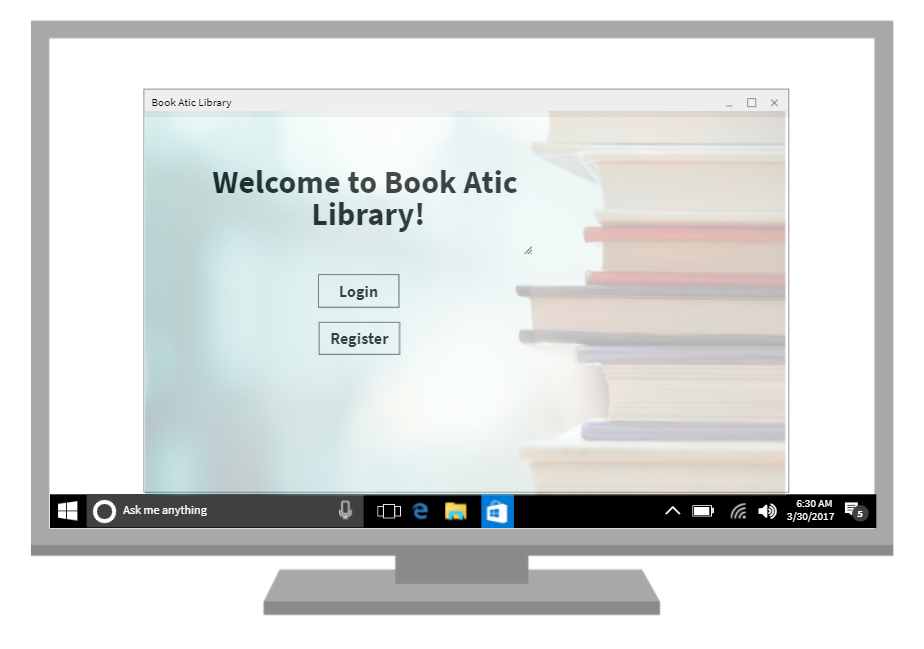
Windows Presentation Foundation (WPF) is a UI framework that creates desktop client applications.

Windows Communication Foundation (WCF) is a framework for building service-oriented applications.

The application guidelines will cover 5 elements:

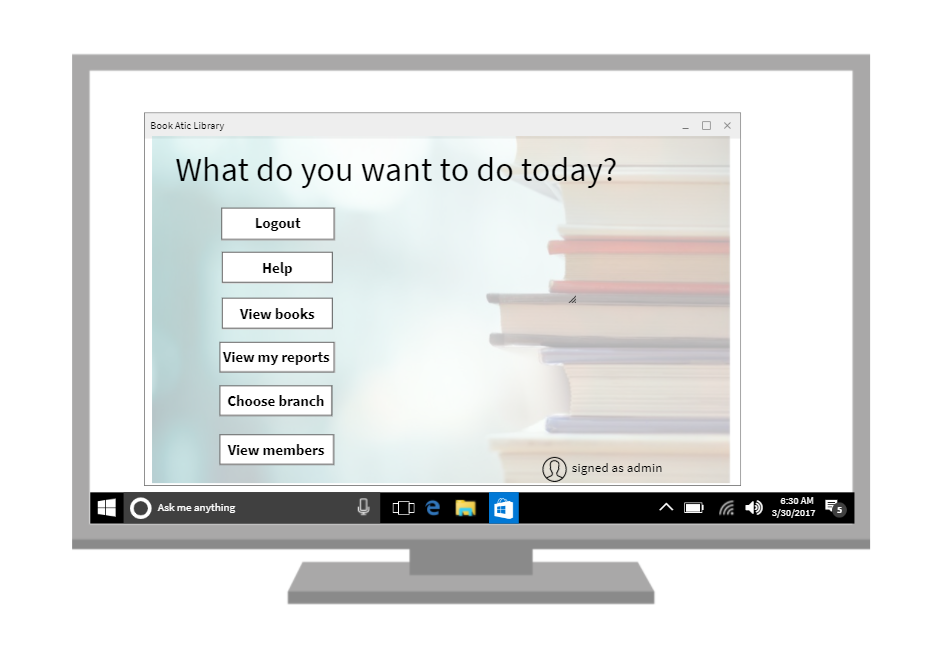
1. Logo
2. Color
3. Typography
4. Illustration
5. Photography

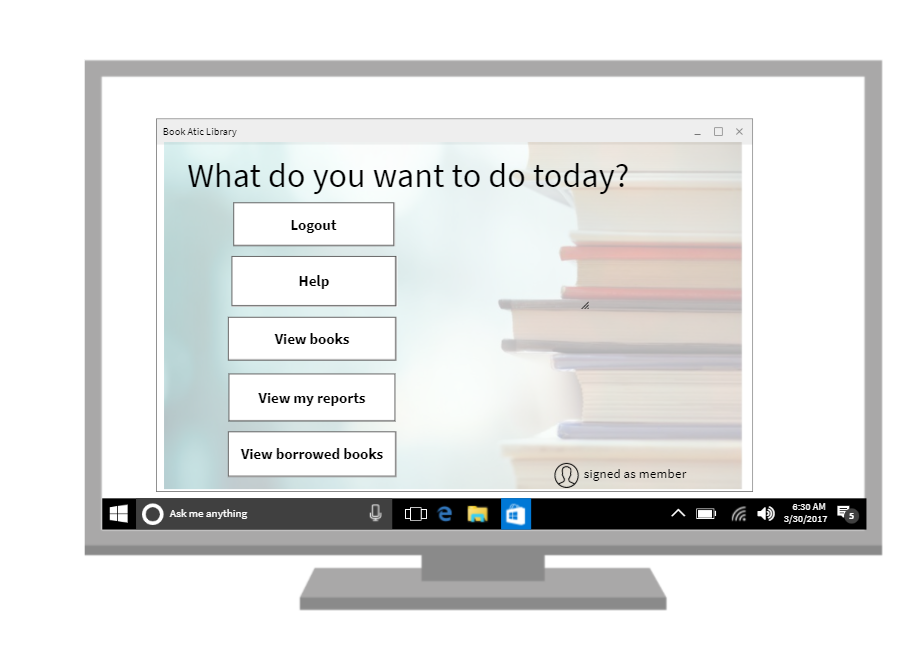
**6.2  Screen Images**

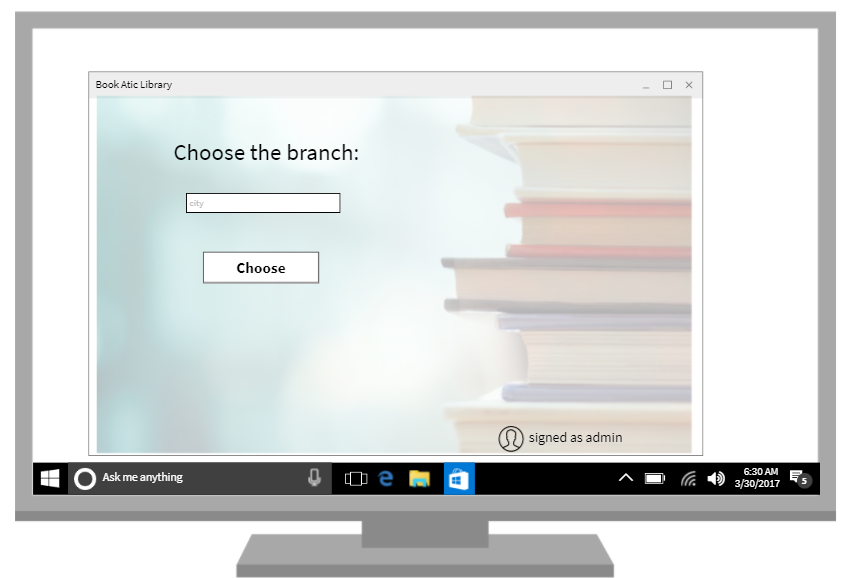
**

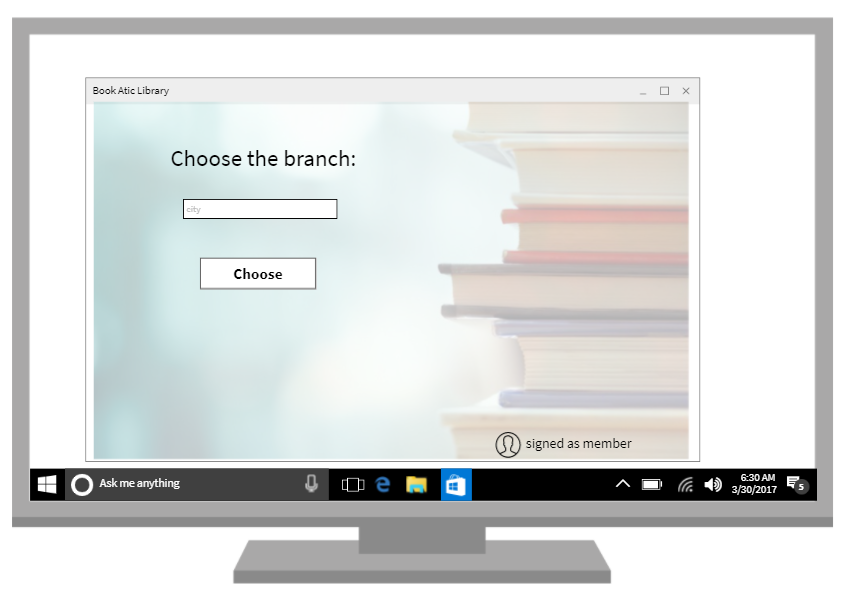
**

**

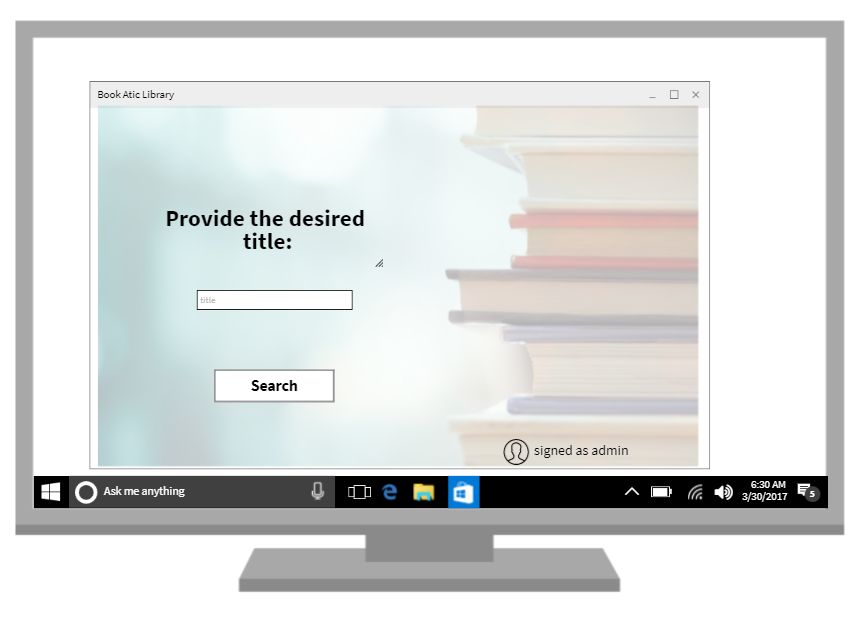
**

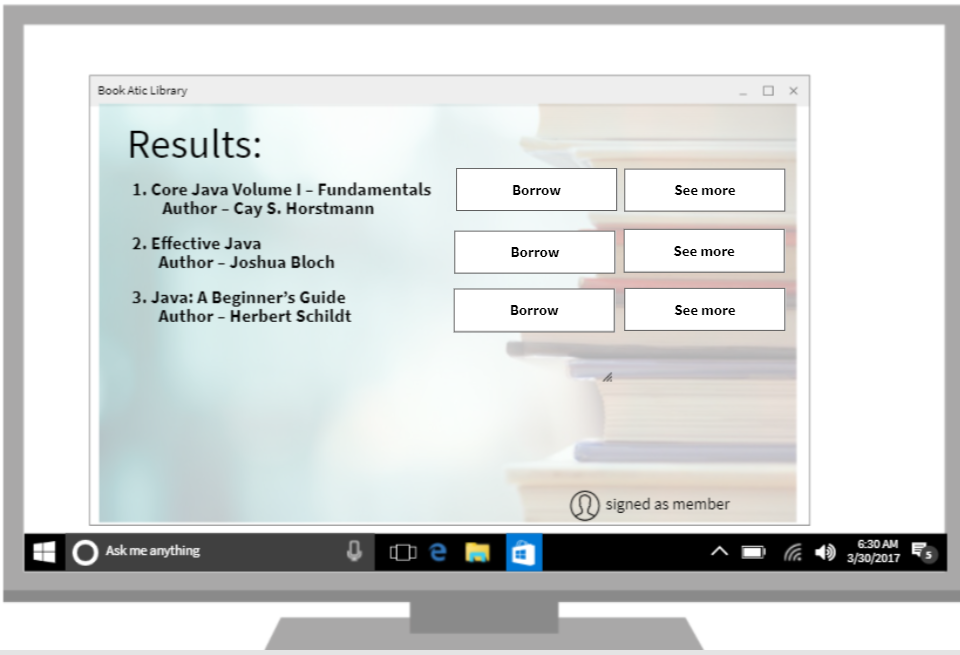
**

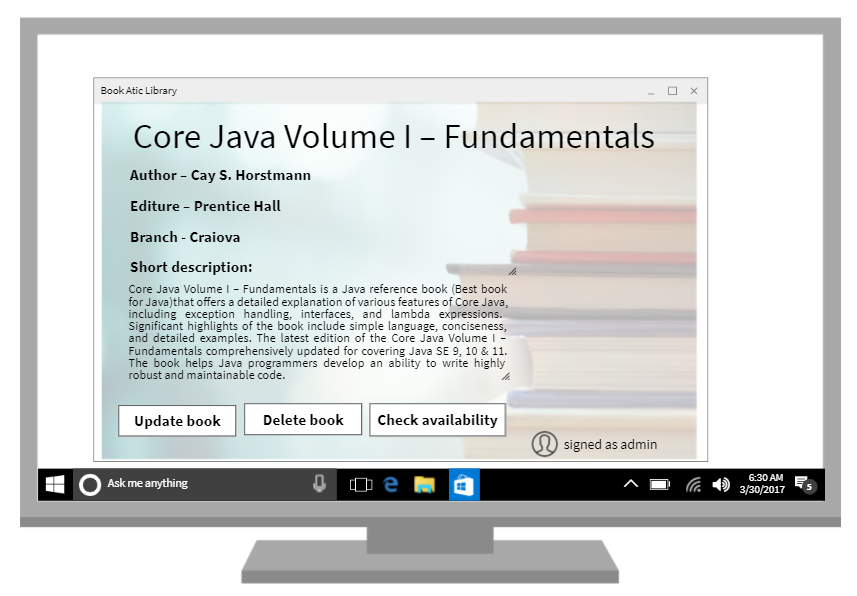
**

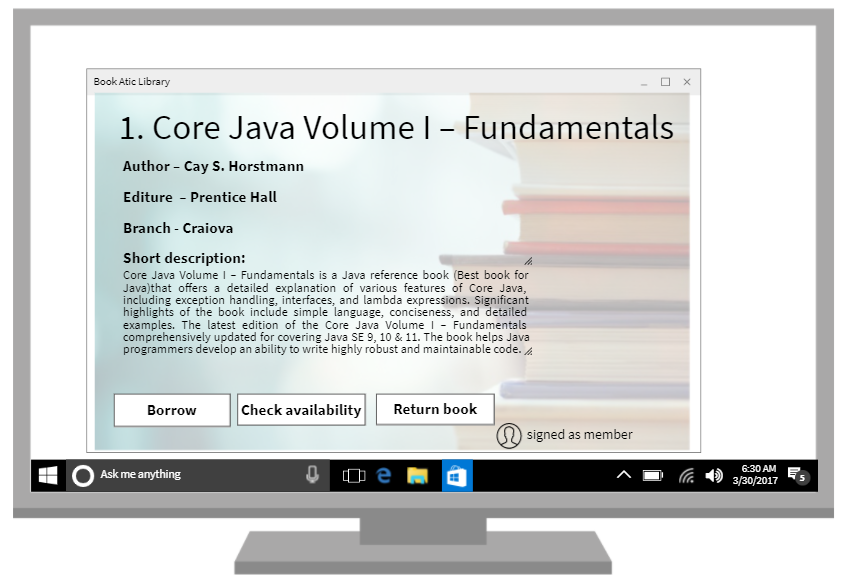
**

**

**

**

**

**

**6.3  Screen Objects and Actions**

A discussion of screen objects and actions associated with those objects.

**7. REQUIREMENTS MATRIX**

Provide a cross reference that traces components and data structures to the requirements in your SRS document.

Use  a  tabular  format to show  which system  components satisfy each of the  functional  requirements from the SRS. Refer to the functional requirements by the numbers/codes that you  gave them in the SRS.

**8. APPENDICES**

*This section is optional.*

4

Software Design Document

Appendices may be included, either directly or by reference, to provide supporting details that could  aid in the understanding of the Software Design Document.

5