**Software Requirements Specification**

**for**

Online Library

**Version 4.2 approved**

**Prepared by**

**Scaunasu Claudia-Nicoleta**

**Ciacaru Cristian-Dumitru**

**Tanasuica Alexandru-Constantin**

**26.02.2021**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

**6. Other Requirements 5**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Scăunașu Claudia | 04.03.2021 | Add User Interfaces and some non-functional requirements | 2.0 |
| Ciacâru Cristian | 04.03.2021 | Complete overall description and add some non-functional requirements | 2.1 |
| Tănăsuică Alexandru | 04.03.2021 | Complete introduction and add some non-functional requirements | 2.2 |
| Scăunașu Claudia | 05.03.2021 | Add features | 3.0 |
| Ciacâru Cristian | 05.03.2021 | Add software interfaces | 3.1 |
| Tănăsuică Alexandru | 05.03.2021 | Add user interfaces | 3.2 |
| Ciacâru Cristian | 08.03.2021 | Finishing features | 4.0 |
| Scăunașu Claudia | 08.03.2021 | Finishing software interfaces | 4.1 |
| Tănăsuică Alexandru | 08.03.2021 | Finishing the document | 4.2 |

# **Introduction**

## **Purpose**

This document describes the implementation details of a software which manages the online library. The product is intended to provide a friendly environment to maintain the details of books, library members and staff. It will also maintain an easy circulation system using computers, provide different reports about members and give the readers’ possibility to borrow books wherever they are. There are two types of users: admin users who can add, update, delete books or check members of the library and reader users that can only borrow books which are available in the library.

SRS describes both functional and nonfunctional requirements of the library system. Some of them are the user interfaces, the design constraints that are to be considered when the system is designed and other factors necessary to provide a complete and comprehensive description of the software requirements.

## **Document Conventions**

→ The entire document should be justified.

→ Convention for main title:

* Font face: Times New Roman
* Font style: Bold
* Font Size: 18

→ Convention for subtitle:

* Font face: Times New Roman
* Font style: Bold
* Font Size: 14

→ Convention for body:

* Font face: Times New Roman
* Font Size: 12

→ Convention for high priority at functionalities:

* Font face: Times New Roman
* Font size: 12
* Font color: red
* Font attribute: italic

→ Convention medium priority at functionalities:

* Font face: Times New Roman
* Font size: 12
* Font color: dark orange
* Font attribute: italic

→ Convention low priority at functionalities:

* Font face: Times New Roman
* Font size: 12
* Font color: dark green
* Font attribute: italic

→ Convention requirements:

* Font face: Times New Roman
* Font size: 12
* Font color: dark blue
* Font attribute: bold

## **Intended Audience and Reading Suggestions**

The document is intended to serve several groups of audiences:

• It is anticipated that the SRS will be used by the application designers. Designers will use the information recorded here as the basis for creating the application's design. They should start by reading the document conventions, product scope, the overall description part, the external interface requirements part, system features, other nonfunctional requirements and other requirements.

• The library manager is expected to review this document. The SRS will serve to establish a basis for agreement between the client and development team about the functionality to be provided by the application. The library manager should watch over product scope(1.4), product perspective (2.1) so he or she could have a starting point for how the website should look like, product functions (2.2), user documentation (2.6), business rules (5.5).

• The application maintainers will review the document to clarify their understanding of what the application does. They should take a closer look at the project just the way designers do so they can build [source code](https://en.wikipedia.org/wiki/Source_code) into a [binary](https://en.wikipedia.org/wiki/Executable) [package](https://en.wikipedia.org/wiki/Software_package_(installation)) for [distribution](https://en.wikipedia.org/wiki/Software_deployment), commit [patches](https://en.wikipedia.org/wiki/Patch_(computing)), or organize code in a [source repository](https://en.wikipedia.org/wiki/Codebase). So, they should start by reading the document conventions, product scope, the overall description part, the external interface requirements part, system features, other nonfunctional requirements and other requirements.

• Testers should also review parts of this document so they can test it and find possible problems which users can encounter. The SRS should picture them the scope and the way one person can use the desktop application. They should be aware of the product scope (1.4) and user documentation (2.6).

• The users which can be of two types: readers or the administrator of the library. The administrator should know the product functions (2.2) and the operating environment (2.4) because he or she can make some modifications within the application that a user cannot and also needs to know on which platforms the product will work. The client should also know on which platforms the project works (2.4). Both of them should know how to use the desktop application, so they need to look over user documentation (2.6).

## **Product 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 will provide the following capabilities:

→ The application will be accessed on a PC at any place.

→ Library staff will be able to manage the book inventory database including remove, change, and add through PC.

→ The application will generate reports for administrative purposes.

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

→ Any new member of the library can register online at any desired time

→ Members can see the availability of a book and can borrow it online, so they only need to go to the library to pick the reserved book

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

## **References**

|  |  |  |
| --- | --- | --- |
| **No. Crt.** | **Short Description** | **URL** |
| 1 | WCF Description | <https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf> |
| 2 | SQL Connection | <https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlconnection?view=dotnet-plat-ext-5.0> |
| 3 | WPF Description | <https://docs.microsoft.com/en-us/dotnet/desktop/wpf/getting-started/walkthrough-my-first-wpf-desktop-application?view=netframeworkdesktop-4.8> |
| 4 | SQL Server | <https://docs.microsoft.com/en-us/sql/ssms/quickstarts/ssms-connect-query-sql-server?view=sql-server-ver15> |
| 5 | C# Tutorial | <https://www.w3schools.com/cs/> |

# **Overall Description**

## **Product Perspective**

Considering that all the existing libraries are web pages, the project will be a new one, developed from scratch, which monitorises the activity of the online library from both members and staff of the library point of view. To the staff of the library admin accounts will be provided so they can track the members and their requests. Any book reader can also have an account as a member and they can see the books which are offered by the library for borrowing.

## **Product Functions**

* Users can have an account in the library
* Users can see the report containing all the borrowed books
* Users can search for the desired book after some criterias (title, author’s name, genre)
* Members can borrow the book
* Members can return the book
* Staff can see the reports for all the members in the library
* Staff can update the reports for all the members in the library
* Staff can add a new book
* Staff can remove a book
* Staff can update a book
* Users can check the availability of the books in the library
* Staff can check if a member did not bring the book in time
* Users can modify his own informations
* Staff can see the informations for all the members

## **User Classes and Characteristics**

We have two classes of users: Staff and Member.

The administrator must work in the library and must have some basic computer skills so he can see the information about members that are registered to the library, check the members, add/update/delete books, update information about both members and themselves.

A member can be anyone who has access to a computer as long as he will be capable of completing simple tasks such as seeing the list of books that are available to borrow. They can also request a book from the staff, and can modify their own information. They will be able to see the books that were borrowed in the past by them and the time frame.

## **Operating Environment**

1. Windows XP
2. Windows Vista
3. Windows 7
4. Windows 8
5. Windows 8.1
6. Windows 10
7. .Net framework v.4.7.2
8. Microsoft SQL Server v.5.7

## **Design and Implementation Constraints**

* The application will be constrained by the capacity of the database.
* We need to be provided with Visual Studio 2019 as an IDE, WPF framework for implementation of the application, as well as xUnit combined to win32mpi for testing the application.

## **User Documentation**

To the final customer we will deliver the application, the database, this document and we will offer online support in case it’s needed.

## **Assumptions and Dependencies**

The basic input devices required for the users are keyboard, mouse and output device is the monitor.

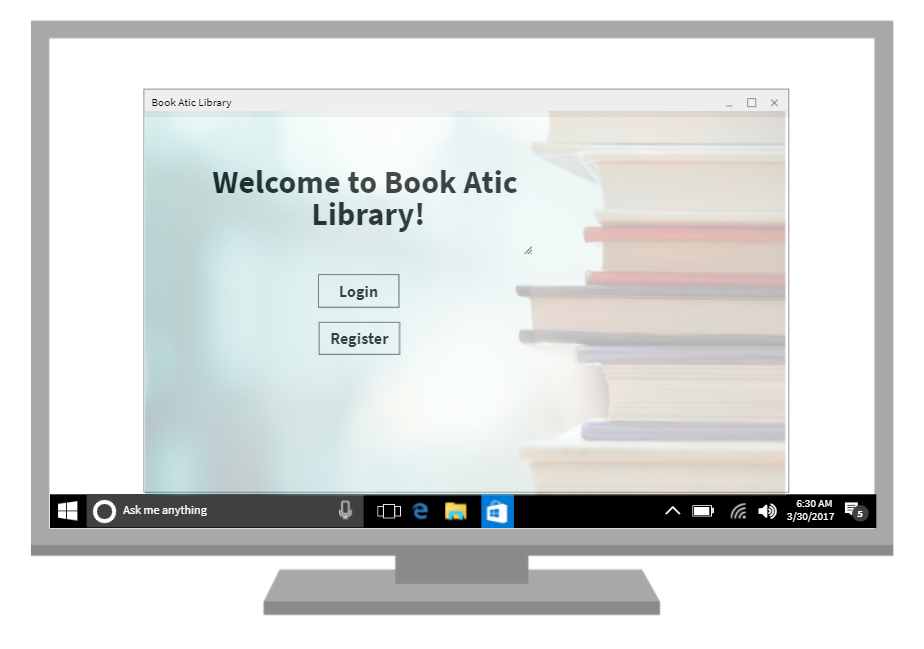
The monitor should have 12” or more and a resolution of at least 800x600.

The application requires the latest version of .Net (v.4.7.2)

For the database is needed Microsoft SQLServer. The user should have his own license for Microsoft SQLServer.

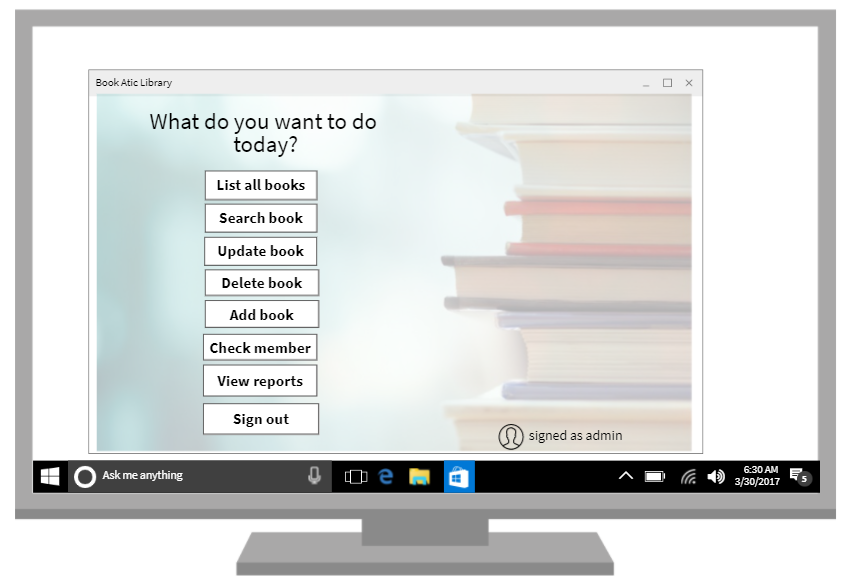
# **External Interface Requirements**

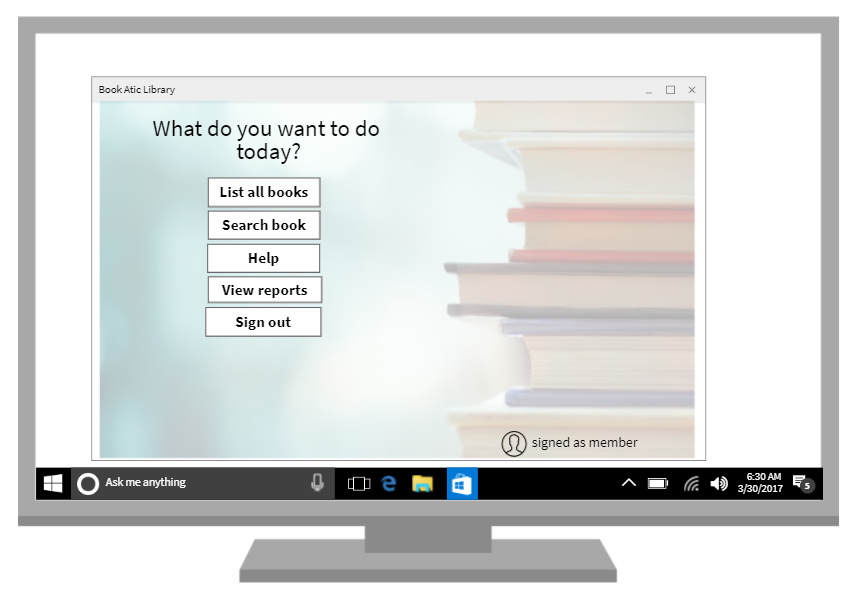
## **User Interfaces**

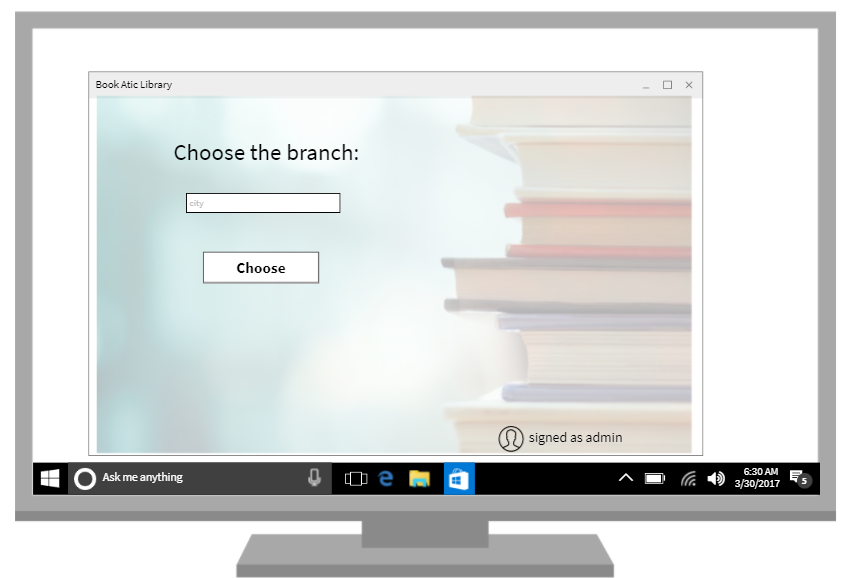
**

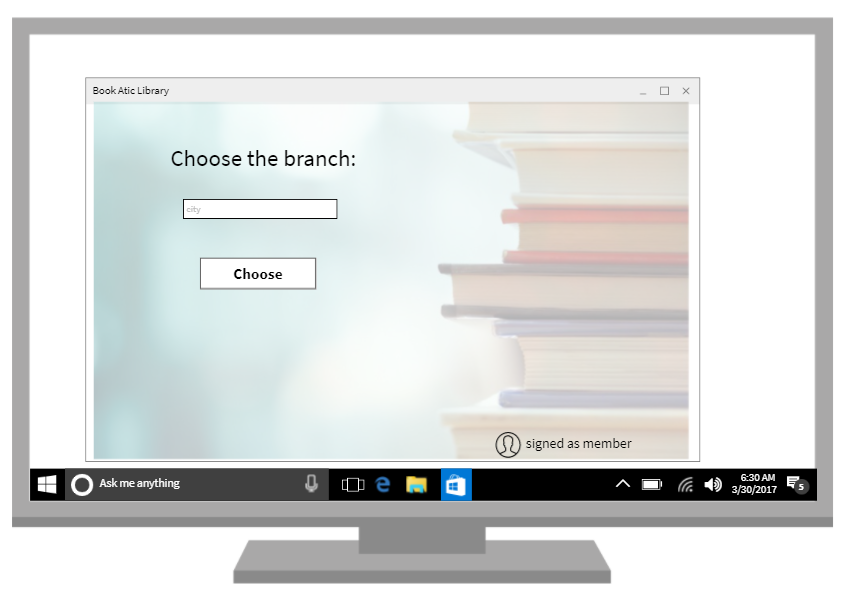
**

**

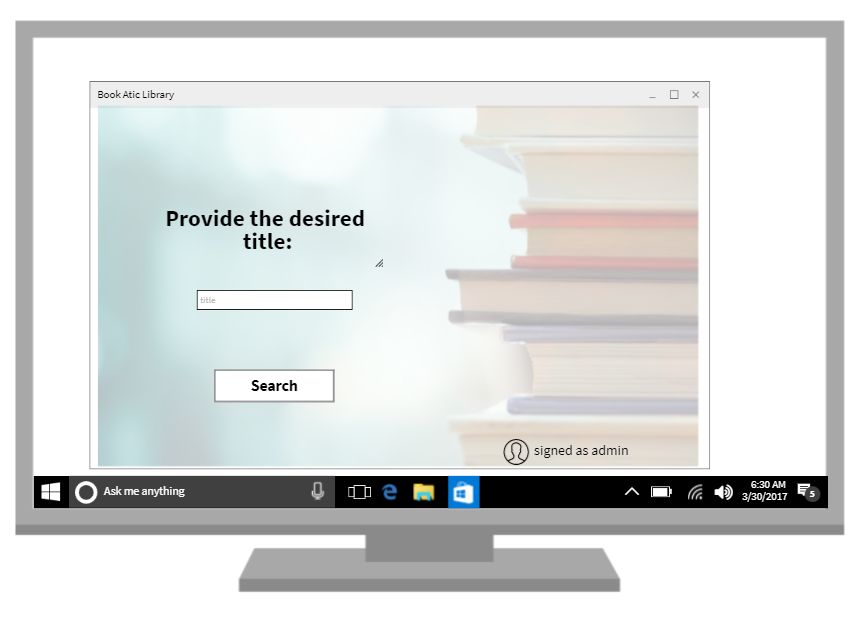
**

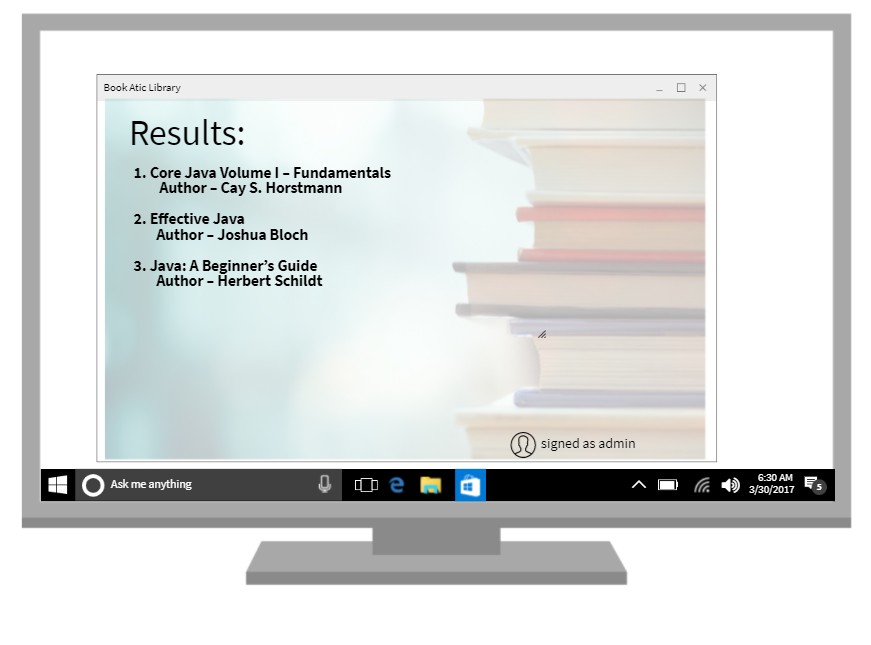
**

**

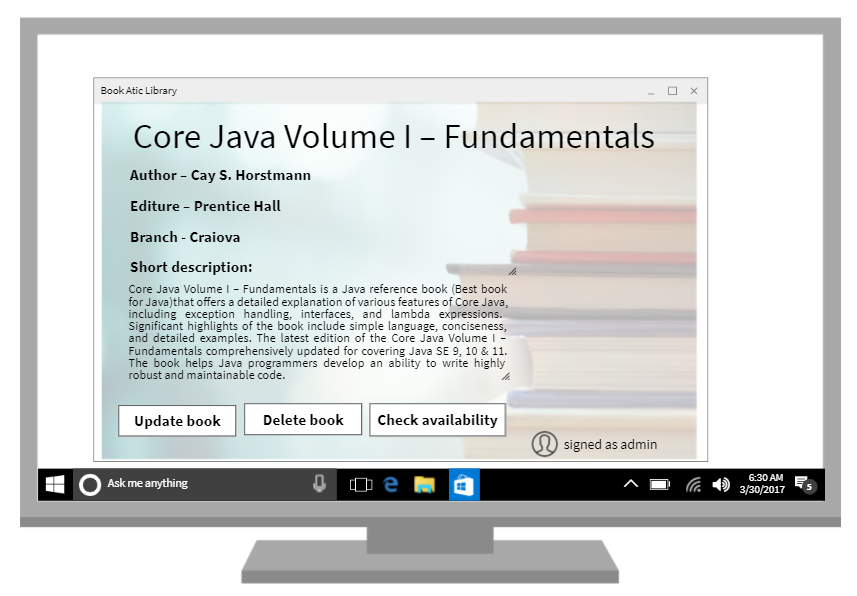
**

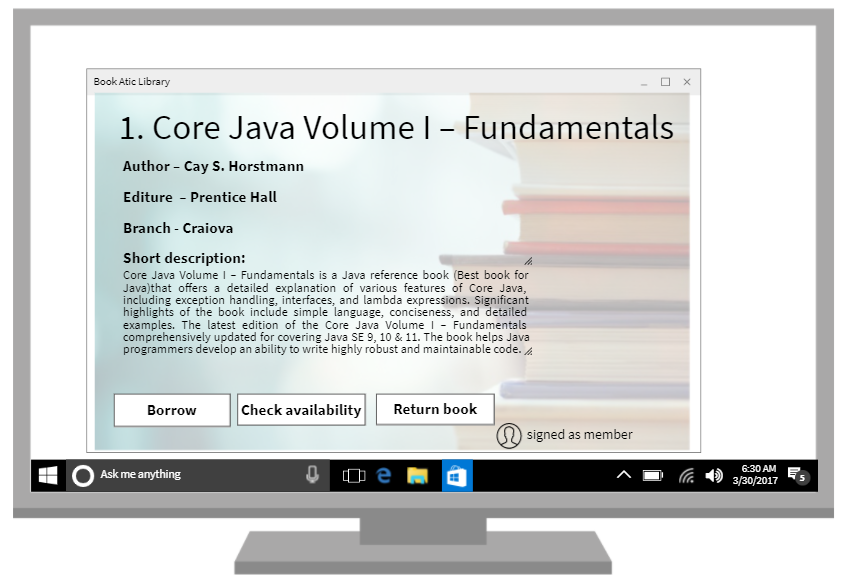
**

**

**

**

**

**

## **Hardware Interfaces**

|  |  |  |
| --- | --- | --- |
| **Minimum specification** | **Required** | **Recommended** |
| *Processor speed (MHz)* | 233 | 300 or higher |
| *RAM (MB)* | 64 | 128 or higher |
| *Free hard disk space (GB)* | 1.5 | >1.5 |
| *Display resolution* | 800x600 | 800x600 or higher |

## **Software Interfaces**

***Microsoft SQL Server*** *is a* [*relational database management system*](https://en.wikipedia.org/wiki/Relational_database_management_system) *developed by* [*Microsoft*](https://en.wikipedia.org/wiki/Microsoft)*.*It will help us to add,update or delete information regarding users or reports.

**Entity Framework** is an open-source [ORM framework](https://en.wikipedia.org/wiki/Object-relational_mapping) for .NET applications supported by Microsoft. It enables developers to work with data using objects of domain specific classes without focusing on the underlying database tables and columns where this data is stored. With the Entity Framework, developers can work at a higher level of abstraction when they deal with data, and can create and maintain data-oriented applications with less code compared with traditional applications.

***Register(first name, last name, username, password, email, age, database)***

input the first name

input the last name

input the username

input the password

input the e-mail

input the age

connect to database

if the username is exists in the database

throw exception

else

add the new account in the database

close the database connection

***Login (username, password)***

input the username

input the password

connect to database

if the username exists in the database

if the password is correct for the username

useroption interface

else

throw exception

else

throw exception

close the database connection

***SearchAfterTitle(bookTitle)***

input the title of the book

choose the branch from a list

connect to database

if the the title of the book exists in the database

show a list with the books

else

throw exception

close the database connection

***SearchAfterAuthorName(authorName)***

input the author’s name of the book

choose the branch from a list

connect to database

if the author’s name of the book exists in the database

show a list with the books

else

throw exception

close the database connection

***SearchAfterGenre(genre)***

input the genre of the book

choose the branch from a list

connect to database

if the genre of the book exists in the database

show a list with the books

else

throw exception

close the database connection

***EditBook(bookName,bookAuthor,bookGenre,bookEditure)***

input new book name

input new book author

input new book genre

input new book editure

connect to database

if any of the fields is empty

throw exception

else

update the book name

update the book author

update the book genre

update the book editure

close the database connection

***AddBook(bookName,bookAuthor,bookGenre,bookEditure)***

input book name

input book author

input book genre

input book editure

connect to database

if any of the fields is empty

throw exception

else

add the book

close the database connection

***DeleteBook(book)***

connect to the database

if the book is not in the database

throw exception

else

delete book from the database

close the database connection

***BorrowBook(book,member)***

connect to the database

if the book is available

if the member is desired

modify the quantity of the book in the database (-1)

write in member’s report the borrowed book

else

throw exception

else

throw exception

close database connection

***ListAllBooks()***

connect to the database

list all the books from the database

close the database connection

***ShowReport(typeOfReport)***

connect to the database

if typeOfReport doesn’t exist in the database

throw exception

else

show report from the database

close the database connection

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.

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, documents, and security.

## **Communications Interfaces**

Not applicable.

# **System Features**

**4.1 Create Account**

*4.1.1* **Description:** The user should be able to register through application using an username and a password

**Priority:** *High*

*4.1.2* **Response Sequences:**

* Users can register by completing some mandatory fields(last name, first name, username, email, password, age). If one of these fields is not completed, the account cannot be created and a message of “Not all the fields were completed” will be displayed.
* If the user is someone who works in the library, the field ID Staff (unique for each person from the library) will be requested. Providing a wrong ID or no ID at all will cause the apparition of a window which will display the message “Wrong ID or unfilled mandatory field” and the account will not be created.
* If an account with that username already exists, a window with a message pops up and informs the user that the chosen username is taken. The user will be asked to provide a new one in the same window.
* If an account with that email address already exists, a window will appear showing that the email address was already registered. In the same window the user will be asked to introduce another one.
* To register, a “Create account” button must be pressed.

*4.1.3* **Functional requirements:**

REQ1: A textbox for each mandatory field must be provided. The mandatory fields are: first name, last name, username, password, email, age. A textbox should be also created for ID Staff

REQ2: It should be checked if the username already exists

REQ3: If it exists, a new account cannot be created, so the program will throw an exception.

REQ4: It should be checked if the email address already exists

REQ5: If it exists, a new account cannot be created, so the program will throw an exception.

REQ6: It should be checked if the password contains at least one capital letter, one digit and at least 6 more characters.

REQ7: If these conditions are not fulfilled, a new account cannot be created, so the program will throw an exception.

REQ8: For staff members it should be checked if the ID Staff exists in the database of the library employees.

REQ9: If it doesn’t exist, a new account cannot be created, so the program will throw an exception.

REQ10: It should be checked if all the mandatory fields are filled. ID Staff field will be completed only if the user is a staff member.

REQ11: If they are not, an exception is thrown.

REQ12: If everything is alright, the application should provide a window with the message “Account created successfully”.

REQ13: The new account which was created will be saved into the database.

**4.2 Login user**

*4.1.1* **Description:** The user should be able to login in application using an username and a password

**Priority:** *High*

*4.1.2* **Response Sequences:**

* Users can login by introducing the username and password. If one of these fields is not completed, the user cannot login and a message of “Not all the fields were completed” will be displayed. A new window with a message will also be displayed if the username or the password is wrong.
* After introducing the username and password the “Sign in” button must be pressed.

*4.1.3* **Functional Requirements:**

REQ14: It must be checked if both username and password are filled in.

REQ15: If one of them (or both) is an empty string, an exception will be thrown.

REQ16: It must be checked if the username and password introduced correspond to the same user.

REQ17: If the username and password do not match for one user, an exception will be thrown.

REQ18: If everything is alright, the user must be redirected to the main page.

**4.3 Logout user**

*4.1.1* **Description:** The user should be able to logout by pressing one button.

**Priority:** *Low*

*4.1.2* **Response Sequences:**

* Users can logout by clicking on the “Sign out” button. The user will be redirected to the starting page.

*4.1.3* **Functional Requirements:**

REQ19: By creating an action for the “Sign out” button, the user currently logged in should be logged out.

**4.4 User’s search functionality**

*4.1.1* **Description:** The user should be able to search a book over some criterias (title, author or genre of the book)

**Priority:** *Medium*

*4.1.2* **Response Sequences:**

* Users should be able choose from a list of branches
* Users can look after a book in the chosen branch by choosing the criteria of searching and then writing the information that they want to find.
* If the criteria is not available for some reason (it does not exist in the database or is misspelled), then a window with the message “Something went wrong” will be displayed.

*4.1.3* **Functional Requirements:**

REQ20: A list of branches should be provided.

REQ21: No search can be done until a branch is selected from the list.

REQ22: A button for each criteria should be provided.

REQ23: In order to choose a criteria, one of the buttons must be pressed.

REQ24: If the “Title” button is clicked, another window will appear in which the user must provide the title of the wanted book.

REQ25: If the field is empty, an exception is thrown.

REQ26: If the field does not exist, an exception is thrown.

REQ27: If the “Author” button is clicked, another window will appear in which the user must provide the author for the wanted book.

REQ28: If the field is empty, an exception is thrown.

REQ29: If the field does not exist, an exception is thrown.

REQ30: If the “Genre” button is clicked, another window will appear in which the user must provide the genre of the wanted book.

REQ31: If the field is empty, an exception is thrown.

REQ32: If the field does not exist, an exception is thrown.

REQ33: If everything is alright, a list of filtered books is displayed

**4.5 User’s view report functionality**

*4.1.1* **Description:** The user should be able to view the reports

**Priority:** *Low*

*4.1.2* **Response Sequences:**

* Members can look at their own reports by choosing which report they want to see.
* Staff users should be able to see all the reports (of members and their own)

*4.1.3* **Functional Requirements:**

REQ34: For staff accounts it should be provided two additional buttons: “Own reports”, “Members reports”.

REQ35: Clicking on one of the buttons mentioned would redirect the staff to another window with buttons for each type of report. The reports are: “Personal information”, “Borrowed books'', “Members integrity”.

REQ36: For member accounts it should be provided three buttons: “Personal information”, “Borrowed books'', “Members integrity”.

REQ37: To be redirected to a specific report, the button for that report must be pressed.

**4.6 Staff user’s search member functionality**

*4.1.1* **Description:** The staff user should be able to search a member by ID registration in the library

**Priority:** *Medium*

*4.1.2* **Response Sequences:**

* Staff users will provide the ID registration of the member and click on the search button.
* If after pressing the search button no member is found another window with the message “Member not found” will pop up.

*4.1.3* **Functional Requirements:**

REQ38: A textbox for ID registration of the member should be provided.

REQ39: Clicking the “Search” button should start the searching of the member.

REQ40: If the textbox is empty an exception is thrown

REQ41: If no result matches an exception is thrown

REQ42: If everything works fine, the staff should be redirected with the page with all that specific user’s reports.

**4.7 Staff user’s check member functionality**

*4.1.1* **Description:** The staff user should be able to see if a member is desired or not

**Priority:** *High*

*4.1.2* **Response Sequences:**

* Staff users will provide the ID registration of the member and click on the check button.
* If no member has that ID, after pressing the check button “Couldn’t check the member” message will be displayed.
* If everything works fine “Desired” or “Undesired” message will pop up

*4.1.3* **Functional Requirements:**

REQ43: A textbox for ID registration of the member should be provided.

REQ44: Clicking the “Search” button should start the searching of the member.

REQ45: If the textbox is empty an exception is thrown

REQ46: If no result matches an exception is thrown

REQ47: If everything works fine, the staff should see the message “Desired”/”Undesired”

**4.8 Staff user’s add book functionality**

*4.1.1* **Description:** The staff user should be able to add books

**Priority:** *High*

*4.1.2* **Response Sequences:**

* Staff users will provide information for all mandatory fields
* Staff users will save the information by clicking on “Add” button
* If one of the fields in not completed, an error message will be displayed
* If the book already exists, an error message will be displayed
* If everything works fine the book will be added

*4.1.3* **Functional Requirements:**

REQ48: A textbox for each mandatory field should be provided.

REQ49: Clicking the “Add” button should add the book in the database if everything works fine.

REQ50: If one textbox is empty, an exception is thrown

REQ51: If the book already exists in the database, an exception is thrown.

**4.9 Staff user’s update book functionality**

*4.1.1* **Description:** The staff user should be able to update books

**Priority:** *Medium*

*4.1.2* **Response Sequences:**

* Staff users will modify whichever number of fields of the book
* Staff users will save the information by clicking on “Modify” button
* If one of the fields in not completed, an error message will be displayed
* If everything works fine the book will be updated

*4.1.3* **Functional Requirements:**

REQ52: A textbox for each field should be provided with the current information in them.

REQ53: Clicking the “Modify” button should update the book in the database if everything works fine.

REQ54: If one textbox is empty, an exception is thrown

.

**4.10 Staff user’s delete book functionality**

*4.1.1* **Description:** The staff user should be able to delete books

**Priority:** *Low*

*4.1.2* **Response Sequences:**

* Staff users will modify press the delete button once they find the book it doesn’t exist in the library
* If everything works fine the book will be deleted

*4.1.3* **Functional Requirements:**

REQ55: A “Delete” button should be provided for each book

REQ56: Clicking the “Delete” button should delete the book and update the database REQ57: A “Book deleted successfully ” message should be displayed

**4.11 User’s check book’s availability functionality**

*4.1.1* **Description:** The users should be able to check the availability of a book.

**Priority:** *High*

*4.1.2* **Response Sequences:**

* Users should be able to press a “Check availability” button
* A message with “Available”/ “Not available” should be displayed

*4.1.3* **Functional Requirements:**

REQ58: A “Check availability” button should be provided for the book

REQ59: Clicking the “Check availability” button should compare the number of that specific book with 0. If it is 0, the book is not available, otherwise the book is available

**4.12 Member user’s renew book functionality**

*4.1.1* **Description:** The member user should be able to renew the date of a borrowed book

**Priority:** *Low*

*4.1.2* **Response Sequences:**

* Member users should be able to press a “Renew” button
* A message with “Renew book was successfully” should be displayed
* If the renew fails for some reason, the “Renew book failed. Please try again” message should be displayed

*4.1.3* **Functional Requirements:**

REQ60: A “Renew” button should be provided for each book

REQ61: Clicking the “Renew” button should automatically add 7 days to the due date of returning the book for that user in the database.

REQ62: If the member does not have the book borrowed, an exception should be thrown

REQ63: A member should be able to renew a borrowed book for a maximum amount of 2 times.

**4.13 Member user’s return book functionality**

*4.1.1* **Description:** The user should be able to return the borrowed book

**Priority:** *Low*

*4.1.2* **Response Sequences:**

* Member users should be able to press a “Return” button
* A message with “You can return the book in the next two days!” should be displayed
* If the returning of the book fails for some reason, the “Returning of the book failed. Please try again” message should be displayed

*4.1.3* **Functional Requirements:**

REQ64: A “Return” button should be provided for each book

REQ65: Clicking the “Return” button should add to the availability of that specific book one unit.

REQ66: If the member does not have the book borrowed, an exception should be thrown

**4.14 User’s edit information functionality**

*4.1.1* **Description:** Users should be able to edit their own information. Additionally, staff users should be able to also edit the users information.

**Priority:** *Medium*

*4.1.2* **Response Sequences:**

* Users should be able to press an “Edit” button
* Member users should be able to edit the information only in the “Personal information” report
* Staff should be able to edit the information in all the reports
* A “Save” button should be provided for the users

*4.1.3* **Functional Requirements:**

REQ67: An “Edit” button should be provided besides each report taking into account what each user can edit.

REQ68: After the information is changed, the “Save” button should update the database

with the new information.

REQ69: If the member tries to update anything else besides the “personal information” report, an exception should be thrown

**4.14 User’s choose branch functionality**

*4.1.1* **Description:** Users should be able to choose the branch in which they are interested

**Priority:** *Medium*

*4.1.2* **Response Sequences:**

* Users should be able to write the branch in which they want to search a book
* If the field is empty or the branch does not exist another window with a “Not found or empty field” will pop up
* A “Choose” button should be provided for the users

*4.1.3* **Functional Requirements:**

REQ70: A textbox for the branch should be provided

REQ71: It should be checked if the field is completed

REQ72: If the field is empty throw exception

REQ73: If the field is not found in the database, throw exception

REQ74: If everything works fine, after the pressing of “Choose” button, the user should be redirected to the page in which he chooses the filter for the search

# **Other Nonfunctional Requirements**

## **Performance Requirements**

1. It refers to the fastness of the search results. The user should see an action as fast as possible (in maximum 2 seconds from the click of the button) so he/she can see the desired results.
2. The application should load and be usable within 3 seconds
3. The application should update the interface on interaction within 2 seconds
4. The database should be updated to at most 1 second after the “save” button is pressed.
5. The system should afford to create a large number of users (10000 users).
6. The admin user should be able to add a new book within 3 minutes
7. Hardware configuration includes Hard Disk: 40 GB

## **Safety Requirements**

Not applicable.

## **Security Requirements**

1. If an admin tries to log in to the app with a non-existing account then the admin should not be logged in. The admin should be notified about log in failure.
2. If a user tries to log in to the app with a non-existing account then the user should not be logged in. The user should be notified about log in failure.
3. If a user wants to create an account and the desired username is used, the user should be asked to choose a different username.
4. If an admin wants to create an account and the desired username is used, the admin should be asked to choose a different username.
5. A user should not be able to login for a certain time period after three times of failed login attempts.
6. An admin should not be able to login for a certain time period after three times of failed login attempts.
7. Normal users can just read information but they cannot edit or modify anything except their personal and some other information.
8. System will have different types of users and every user has access constraints.
9. When creating an account the password should have at least one capital letter, one digit and the size of the password to be at least 6.
10. When typing the password no one should be able to see it, so instead of the actual password on the screen will appear “\*” for each character or digit.

## **Software Quality Attributes**

***5.4.1 Quality code rules***

1. Write only one statement per line
2. Write only one declaration per line
3. Use implicit typing for local variables when the type of the variable is obvious from the right side of the assignment, or when the precise type is not important Use a try-catch statement for most exception handling
4. Use object initializers to simplify object creation.
5. Use meaningful names for query variables.
6. Use implicit typing in the declaration of query variables and range variables.
7. Each “if” statement should have an else statement.
8. Use aliases to make sure that property names of anonymous types are correctly capitalized, using Pascal casing.
9. Use KISS (keep it simple, stupid)
10. Use DRY (Don’t repeat yourself) - avoid duplication.
11. Use SOLID principle
12. Programs should be separated into distinct sections, each addressing a separate concern, or set of information that affects the program.
13. Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification
14. Subtypes must be substitutable for their base types.
15. Nulls are not substitutable for actual instances
16. Prefer small, cohesive interfaces to large ones
17. NotImplementedException's should be addressed / implemented
18. Abstractions should not depend on details. Details should depend on abstractions
19. A method should have a single purpose.
20. A method should use all provided parameters.
21. A method should either change the state of the object or return a value, not both.

***5.4.2 Functional quality rules***

1. The search feature will be provided through a “Search” button which will be intuitive and easy to find for the user.
2. The different search options will be obvious, simple and easy to understand since buttons will be provided for each filter in order for the user to perform book search easily.
3. The results will be displayed in the list view in order to be user friendly and easy to understand.
4. Selecting an element in the result list should only take one click so that the user can use the listview easily.
5. The system should continue to work *correctly* (performing the correct function at the desired level of performance) even in the face of *adversity* (hardware or software faults, and even human error)
6. Sustain scalability through storing the data about books, users (both members and staff), reports, so all this information can be found and used later.
7. Providing good documentation and an easy-to-understand operational model (help button from UI)
8. Providing good default behavior, but also giving administrators the freedom to override default informations when needed(the administrator should be able to modify informations about books and members)

## **Business Rules**

Not applicable.

# **Other Requirements**

Not applicable.

**Appendix A: Glossary**

|  |  |  |
| --- | --- | --- |
| **No. Crt.** | **Abbreviation** | **Meaning** |
| 1 | IDE | Integrated Development Environment |
| 2 | SRS | Software Requirement Specification |
| 3 | UI | User Interface |
| 4 | WPF | Windows Presentation Foundation |
| 5 | WCF | Windows Communication Foundation |
| 6 | IIS | Internet Information Services |
| 7 | XML | eXtensible Markup Language |
| 8 | ORM | Object–relational mapping |

**Appendix B: Analysis Models**

Not applicable.

**Appendix C: To Be Determined List**

Not applicable.