

**Library Management System**  
**Software Requirements Specification**  
**Version 1.0**

Prepared for  
BSU-COMPSCI 471—Software Engineering  
Instructor: Jim Buffenbarger  
Fall 2012

A. Benz      Q. Bowers      S. Kausler      J. Kress

12/11/2012

## Revision History

<i>Date</i>	<i>Description</i>	<i>Author</i>	<i>Comments</i>
12/11/2012	Version 1.0	Allen Benz Quincy Bowers Scott Kausler James Kress	This is the first revision of the specification.

## Document Approval

<i>Signature</i>	<i>Printed Name</i>	<i>Title</i>	<i>Date</i>
	Allen Benz	Software Engineer	12/11/12
	Quincy Bowers	Software Engineer	12/11/12
	Scott Kausler	Software Engineer	12/11/12
	James Kress	Software Engineer	12/11/12

## Table of Contents

<b>Revision History</b>	<b>ii</b>
<b>Document Approval</b>	<b>ii</b>
<b>List of Figures</b>	<b>vi</b>
<b>1 General Description</b>	<b>1</b>
1.1 Executive Summary . . . . .	1
1.2 Feature Description . . . . .	2
1.2.1 Book Collection . . . . .	3
1.2.2 Member Database . . . . .	4
1.2.3 Interface to Update the Collection . . . . .	4
1.2.4 Updating the Member Database . . . . .	5
1.2.5 Searching the Collection . . . . .	5
1.2.6 Check-Out Books . . . . .	6
1.2.7 Check-In Books . . . . .	7
1.2.8 Imposing a Fee . . . . .	7
1.2.9 Renew a book . . . . .	7
1.2.10 Reserve a Book . . . . .	8
1.2.11 Barcoded Books . . . . .	8
1.2.12 Member ID Cards . . . . .	8
1.3 User Characteristics . . . . .	8
1.4 Environmental and Social Impact . . . . .	10
1.4.1 Environmental Impacts . . . . .	10
1.4.2 Social Impacts . . . . .	10
1.5 Perspective and Scope . . . . .	10
1.5.1 System Perspective . . . . .	10
1.5.2 Scope . . . . .	11
1.6 Assumptions, Dependencies and Constraints . . . . .	11
1.6.1 Assumptions . . . . .	11
1.6.2 Dependencies . . . . .	11
1.6.3 Constraints . . . . .	11
1.7 Standards Compliance . . . . .	11
1.7.1 Software Standards . . . . .	12
1.7.2 Hardware Standards . . . . .	12
1.7.3 Reuse of Existing Software . . . . .	12

<b>2</b>	<b>Functional Requirements</b>	<b>13</b>
2.1	Feature Description . . . . .	13
2.1.1	Calculating Book Reservations . . . . .	13
2.1.2	Renewing Fees . . . . .	14
2.1.3	Calculating UUIDs . . . . .	14
2.2	Use Cases . . . . .	14
2.2.1	Check Out a Book . . . . .	14
2.2.2	Check in a Book . . . . .	15
2.2.3	Reserve a Book . . . . .	16
2.2.4	Book Search . . . . .	17
2.2.5	In-Library Member Search . . . . .	17
2.2.6	Adding a Book to the <a href="#">Collection</a> . . . . .	18
2.2.7	Updating a Book . . . . .	18
2.2.8	Deleting a Book from the <a href="#">Collection</a> . . . . .	19
2.2.9	Adding a <a href="#">Member</a> . . . . .	19
2.2.10	Updating a <a href="#">Member</a> . . . . .	20
2.2.11	Deleting a <a href="#">Member</a> . . . . .	20
2.2.12	Pay Fees . . . . .	21
2.2.13	Renew Book . . . . .	21
2.2.14	Failed Check Out . . . . .	22
2.2.15	Failed Search . . . . .	22
2.2.16	Failed In-Library Member Search . . . . .	23
2.2.17	Failed Adding a Book to the <a href="#">Collection</a> . . . . .	23
2.2.18	Failed Updating a Book . . . . .	24
2.2.19	Failed Adding a <a href="#">Member</a> . . . . .	24
2.2.20	Failed Updating a <a href="#">Member</a> . . . . .	25
2.2.21	Failed Deleting a <a href="#">Member</a> . . . . .	25
2.2.22	Cancel Action . . . . .	26
2.3	User Interfaces . . . . .	26
2.3.1	<a href="#">Library Interface</a> . . . . .	26
2.3.2	Web Interface . . . . .	27
2.4	Hardware Interfaces . . . . .	33
2.5	Software Interfaces . . . . .	33
<b>3</b>	<b>Nonfunctional Requirements</b>	<b>34</b>
3.1	Time and Space Performance . . . . .	34
3.2	Reliability . . . . .	34
3.3	Portability . . . . .	35
3.4	Security . . . . .	36

<b>4 System Tests</b>	<b>37</b>
4.1 Test Time and Coverage Estimate . . . . .	37
<b>Glossary</b>	<b>38</b>
<b>Appendix A</b>	<b>40</b>
<b>References</b>	<b>41</b>

## List of Figures

1	The Primary screen. . . . .	28
2	The first screen of the <a href="#">Update Collection Interface</a> . . . . .	28
3	The first screen of the <a href="#">Update Members Interface</a> . . . . .	29
4	The Book Search screen. . . . .	29
5	The Member Search screen. . . . .	30
6	The Edit Book screen. . . . .	30
7	The Edit Member screen. . . . .	31
8	Several example Confirmation screens. . . . .	31
9	The Web Search screen. . . . .	32
10	The Book Information screen. . . . .	32
11	Dataflow Diagram . . . . .	40

# 1 General Description

This document describes an integrated library system. The system, henceforth to be referred to as the System, consists of the software described in this document and the hardware on which it will run. Specifications were taken from the Library of Congress standards[1]. We based the System on basic features described in an open source library management system[2].

## 1.1 Executive Summary

This document covers the requirements and analysis of a library management system. The System is based on a database of books and a database of library **members**. Our goals for the System are to make it user friendly and easy to use.

Books are identified by a unique key and a Library of Congress call number [1]. Other identifiers include title, description, author, edition, year published, current state, type, and due date. The database of **members** includes basic identifying information for library **members** as well as lists of currently checked out books, reserved books, and outstanding fees.

There are two ways to access the System. The first is only available to **librarians** in the library. Librarians have the ability to check in, check out, and renew books on behalf of **members**. They may also create, edit, or delete book and **member** entries from the database. They may also use the System to search for books. The second way to access the System is from an online interface. Users can use the search online via the library's home page.

Because of the librarian's persona, the System should feature an easy to use interface with clear instructions for each task. Because **patrons** want a fast way to search for books and the ability to narrow down the online search, the search function should use keywords but should also have advanced search functionality.

The System has minimal environmental impacts. In fact, it may have positive environmental impacts by reducing the library's use of paper. Social impacts are also minimal. Only limited information about a library **member** is stored and only the **librarian** has access to that information.

We assume that only **librarians** can make changes to the database through the System and the System will run on off-the-shelf computer hardware. There are no dependencies. The one constraint is the System only works with a single **collection**.

There are 22 use cases for this system. Some examples including checking in a book, checking out a book, renewing a book, modifying books in the **collection**, modifying library **members**, and searching for books. A **patron** may be involved in a book search but only the **librarian** is involved in the rest of the use cases.

The **Library Interface** is part of the user interface for the System and consists of

eight main screens.

- A Primary screen that gives six menu options to perform several different actions.
- An [Update Collection Interface](#) screen has three buttons that allow the [librarian](#) to add a book, search for a book, or cancel.
- An [Update Members Interface](#) screen has three buttons that allow the [librarian](#) to add a [member](#), search for a [member](#), or cancel.
- A Book Search screen that has search fields, a **Search** button, search results, a **Cancel** button, and a **Search Book** button.
- A [Member](#) Search screen that has search fields, a **Search** button, search results, a **Cancel** button, and a **Search Member** button.
- An Edit Book screen allows the [librarian](#) to change the fields associated with a book.
- An Edit [Member](#) screen allows the [librarian](#) to change the fields associated with a [member](#).
- A Confirmation is used to confirm an action or inform the [librarian](#) that an action has been successfully (or unsuccessfully) performed.

The [Web Interface](#) is another part of the user interface that includes the following.

- A Web Search screen that includes search fields and search results.
- A Book Information screen that displays information about a selected book.

The hardware interface for the System consists of a standard desktop computer and a barcode scanner.

The software interface for the [Library Interface](#) interacts with the background database. The [Web Interface](#) interacts with a Web server and the background database.

The [Library Interface](#) for the System must be reliable 99% of the time, meaning it can only not work 1% of the time. This depends on the reliability of the library's computer system and the database. The [Web Interface](#) must also be reliable 99% of the time. That depends on the Web server and the [Library Interface](#).

The [Library Interface](#) will be portable to Linux and Windows. The [Web Interface](#) is portable by its nature, but should be tested on several different Web browsers.

## 1.2 Feature Description

This section will describe the primary features of the System. These features will include the book [collection](#) database, the member database, the user interfaces to the in-library and Web features, and the procedures for updating and searching both the member and [collection](#) databases.



### 1.2.1 Book Collection

The System will maintain a database of books which the library owns. This is the electronic representation of the library's [collection](#). The database will have a single entry for each book in the [collection](#). Each entry will consist of the following fields.

- [Universally Unique Identifier \(UUID\)](#) - A unique identification number which identifies a single physical book in the [collection](#).
- [Library of Congress Classification \(LCC\)](#) - A Library of Congress call number which identifies a given printing of a book.
- [International Standard Book Number \(ISBN\)](#) - A numbering system which uniquely identifies books for commercial purposes.
- Title - The title of the book.
- Subject - Keywords describing what subject matter the book contains.
- Description - A brief description of the book.
- Author - The author(s) of the book.
- Edition - Identifies the edition of the book.
- Year - The year the book was published.
- Status - One or more of the following possible states:
  - checked in (mutually exclusive with checked out and restricted states.)
  - checked out (mutually exclusive with checked in and restricted states.)
  - [restricted](#) (mutually exclusive with checked in and checked out states.)
  - [reserved](#) - A book is in this state if a member has reserved the book. *See Reserve a Book 1.2.10*
  - [damaged](#) - Denotes physical damage to the book.
- Type - Fiction, Reference, etc.
- Due Date - The date the book is due back if it is currently checked out.
- Value - The monetary value of the book. This field is also the maximum fee that a member would have to pay if the book is not returned on time.
- Reservation Count - The number of members who have reserved the book.

When new books are added to the [collection](#) a barcoded label will be printed which can then be affixed to the back of the book for easy input of the book's [UUID](#) into the System's interfaces.

### 1.2.2 Member Database

The System will maintain a database of library **members**. A library **member** is allowed to borrow books from the library. A database entry for a library **member** consists of the following fields:

- **UUID** - A unique identification number which identifies a single library **member**.
- First Name
- Last Name
- Mailing Address
- Phone Number
- List of Currently Borrowed Books
  - Each entry in this list contains the following three fields:
    - \* The book's **UUID**.
    - \* The date the book was checked out.
    - \* The date the book is due to be returned.
- List of Reserved Books
  - Each entry in this list contains the following three fields:
    - \* The book's **UUID**.
    - \* The date the book is due to be returned.
    - \* The number of people who reserved the book before this member.
- Outstanding Fees

When new members are added to the database a barcoded ID card will be printed which is given to the member. This can then be used to quickly bring up the member's account information while checking out books and paying fines.

### 1.2.3 Interface to Update the Collection

The System needs to have an interface which will allow a **librarian** to update the **collection**. They need to be able to add new books to the **collection**, remove books from the **collection**, and modify existing entries in the collection.

The system must ensure that an entry cannot be added which has the same **UUID** as an existing entry.

Only **librarians** can access this interface.

The **librarian** can access an existing book's entry by executing a search on any of the following fields.

- [UUID](#)
- [ISBN](#)
- Title
- Subject
- Description
- Author

#### 1.2.4 Updating the Member Database

The System needs to have an interface which will allow a [librarian](#) to update the database of library members. Members can be created, modified, or removed.

The System must ensure that a member entry cannot be added which has the same [UUID](#) as an existing member entry.

Only [librarians](#) can access this interface.

All fields in a [member](#) entry are modifiable except for the [UUID](#), which is assigned automatically when the [member](#) is added and never re-assigned.

The [librarian](#) can access an existing member's entry by executing a search on any of the following [member](#) fields.

- [UUID](#)
- First Name
- Last Name
- Mailing Address
- Phone Number

#### 1.2.5 Searching the Collection

The System needs an interface that will allow any user of the System to search for a book in the [collection](#).

The [collection](#) can be searched using a standard keyword search mechanism that will return a list of the books in the [collection](#) whose database fields match at least some of the specified keywords.

The fields that are searched in the database are:

- [ISBN](#)
- [UUID](#)

- Title
- Subject
- Description
- Author

The [patron](#) can execute a search on specific fields, or they can search across all fields at once. The list of search results can be sorted alphabetically by title or author as well as by [relevance](#) in ascending or descending order. By default the results are sorted by [relevance](#).

The search results will include the following fields for each item returned.

- UUID
- [LCC](#)
- [ISBN](#)
- Title
- Subject
- Description
- Author
- Edition
- Year
- Status
- Type
- Due Date

### 1.2.6 Check-Out Books

[Members](#) may borrow a book from the library. Only a [librarian](#) can check out a book to a member. Non-members cannot borrow books. When a book is checked out the status field in its database entry is changed to *checked out* and the Due Date field is updated as described below.

There are several conditions that may prevent a book from being checked out.

1. The book has a status of *restricted*.

2. The book has a status of *reserved* and the reserving member is not the member who is attempting to borrow the book.
3. The member who is attempting to borrow the book has an unpaid fee.
4. The member has already checked out the maximum number of books.
  - A member can check out at most 100 books simultaneously.

A book may be checked out for a period of time equal to the **loan period**. The **loan period** is 28 days long and the day after the book is checked out is counted as the first day. A book that is checked out on the 1st of the month, is therefore due back on the 29th of the same month.

### 1.2.7 Check-In Books

When a **member** returns a checked-out book a **librarian** must check it back into the System before it can be put back onto a shelf. When a **librarian** checks in a book the status field is updated to *checked in* and the due date field is set to a null value.

### 1.2.8 Imposing a Fee

If a member fails to return a book to the library by the due date they are subject to a fee. The fee increases on a schedule as long as the book remains checked out.

The fee schedule is \$0.25 for each day the book is not checked in past the due date. Fees are imposed automatically by the System everyday at midnight.

As an example, if a book is due back on the 10th but is not checked in until the 15th, the fee will be \$0.25 for the 11th, 12th, 13th, and 14th, for a total of \$1.00.

### 1.2.9 Renew a book

The System needs an interface to renew a book that is currently checked out to a member.

When a book is renewed the Due Date is extended by the amount of the **loan period**.

An attempt to renew a book can fail under the following conditions.

1. The book has a status of *reserved*.
2. The member who is attempting to renew the book has an unpaid fee.

A book can be renewed indefinitely as long as either of the above two conditions are never met.

### 1.2.10 Reserve a Book

The System allows [members](#) to reserve books. This may also be referred to as putting a hold on a book.

If a member wants to ensure the availability of a book for checkout they can reserve it. This guarantees that the book will be available at a specific time for the member to borrow it. If a book is currently checked-in to the library when a member reserves it, all other members are immediately prohibited from borrowing the book. A [librarian](#) contacts the member and informs them the book has been held for them. The [librarian](#) should then place the book in a special place in the library so that it can easily and quickly be found for the member when they come in to borrow it.

If the book is currently checked out when a reserve is placed on it, then the book will no longer be eligible to be renewed and no other members will be able to borrow the book when it is returned. The book is placed in a special area in the library on being returned so that it can easily and quickly be located for the reserving member.

If a book has already been reserved by one or more [members](#), any new reservations are placed into a queue. The [member](#) at the front of the queue will have exclusive rights to borrow the book when the book is checked in.

When a reserved book is checked out by the member at the front of the reservation queue, that member is removed from the reservation queue. If a member attempts to place a second reservation on the same book, the new reservation fails.

### 1.2.11 Barcoded Books

Books in the library's [collection](#) will have a label fixed to the back of the book which will have a barcode representation of the book's [UUID](#) as well as the numeric [UUID](#) printed on it. This can be used to quickly bring up the book's database entry in the System, either by manual entry of the UUID or by scanning the barcode with a barcode scanner, which will be present at each terminal.

### 1.2.12 Member ID Cards

[Members](#) will receive an ID card. The ID card will have the member's name and [UUID](#) printed on it as well as a barcode representation of the [UUID](#). When checking out a book, the member can present their member ID card which the [librarian](#) can then scan, with the barcode scanner, to quickly pull up their account.

## 1.3 User Characteristics

There are two main categories of users for the System. The first is the librarian and the second is the [patron](#).

In order to best determine the characteristics for a [librarian](#), we will list certain traits a [librarian](#) is likely to possess. The [librarian](#) is not an advanced computer user. He or she is likely working in the library because of a love of books. The [librarian](#) prefers a very structured life and works best with clear procedures. A [librarian](#) may be elderly and can have vision problems (as well as arthritis). A [librarian](#) may also be a volunteer.

A typical day for the [librarian](#) may be waking up early in the morning. The [librarian](#) probably spends time reading in the morning before going to the library. At the library, the [librarian](#) spends a majority of his or her time shelving books and helping library [patrons](#). He or she will leave the checkout terminal and come back many times over the course of the day.

Most of the time, the [librarian](#) will check out books for a [member](#) or search for books in the System. However, the [librarian](#) occasionally will be updating the [member](#) and [collection](#) databases. The [librarian](#) may also need to train volunteers or new employees on how to use the System. When the [librarian](#) is checking books in, he or she prefers to check in multiple books at the same to increase efficiency.

Because of the librarian's characteristics, we want to make the System as easy to use as possible. This means that we want large text and big buttons along with clear instructions on how to do everything. The System should be as automated as possible. Since [librarians](#) may know little to nothing about computers, we should favor multiple screens over scrolling screens for tasks that use the whole screen. Since the [librarian](#) is coming back and forth to the terminal, it should be easy to start a new task on the System. Because the [librarian](#) likes things to be structured, no confusing errors should ever pop up and each step in the System should be clear. [Librarians](#) should be able to use the System with minimal training.

A [patron](#) can be any person who visits the library's Website to search for books. Since the [patron](#) feels comfortable using the Internet to search for books over calling or visiting the library, he or she likely has at least an elementary knowledge of computers. However, the [patron](#) may be busy and does not want to spend too much time optimizing the search for the right book. Because of that, the search should be based around a keyword search, but it should have advanced features for [patrons](#) who want to better narrow their search. If it is not readily available on the the library's homepage, a link to the search page should clearly be provided.

A [patron](#) can be anyone, from a college student who has used computers his or her whole life, to someone who is new to computers. Because of that, it is hard to determine the typical characteristics for a [patron](#). The online search interface should be easy to use for those who want to casually browse books, but it should have many advanced options for [patrons](#) who want to search for something specific. For example, advanced [patrons](#) may want to be able to search for a journal article on a certain subject. Books should be searchable by simply typing in some keywords and clicking the **Search** button and

advanced features should be readily accessible.

## **1.4 Environmental and Social Impact**

The following is an overview of the potential environmental and social impacts of the System, specifically relating to [librarians](#) and library patrons.

### **1.4.1 Environmental Impacts**

The environmental impacts of the System are likely to be negligible in nature. The design of the System is such that its installation in the library will either maintain the status quo or improve it. If the library is currently using an online or computer based system for managing their [collection](#), this software will not modify any current environmental effects. However, if the library is using a ledger approach to manage their [collection](#), this system will have positive environmental effects. It will allow the library to reduce its use of paper, and make management of the [collection](#) more efficient and cost effective.

### **1.4.2 Social Impacts**

The design and layout of the System system is such that social impacts of the System should be minimal. Only necessary information about a [member](#) will be stored on the System at any given time. Stored information that could affect a [member](#) in a negative way include first and last name, mailing address, phone number, currently checked out books, currently reserved books, and unpaid fees. The System purposefully limits information kept about any given [member](#) in order to protect personal privacy. As such, information that is kept will only be available to [librarians](#) physically at the library. No online or network access to the [member](#) database will be possible.

No other social or environmental impacts resulting from or due in part to this system are anticipated.

## **1.5 Perspective and Scope**

This section details the perceived perspective and scope of the System.

### **1.5.1 System Perspective**

The System is intended to be used by a single library managing a single collection. The System will allow [librarians](#) to easily manage the [collection](#) of their library through book searches, book check in, book check out, [member](#) database management, and [collection](#) database management. The System allows users to perform searches for books in the [collection](#). The System is intended to run on both the Windows and Linux operating



systems in order to provide the greatest flexibility. In short, the System is a highly flexible and intuitive collection management system for the product conscious consumer.

### **1.5.2 Scope**

The scope of the System is a single library managing a single [collection](#). The System does not support inter-library loans or book searches of other libraries' collections. This allows the System to be self-contained and easy for a library to implement and use. Very little training or computer skills will be required to effectively operate the System.

## **1.6 Assumptions, Dependencies and Constraints**

### **1.6.1 Assumptions**

The following assumptions are made about the System and the environment in which it will operate.

Only [librarians](#) will have the ability to make changes to the databases in the System. All other users will only be able to execute searches to find books.

The System will run on off-the-shelf computer hardware.

### **1.6.2 Dependencies**

The successful operation of the System depends on the following.

- Electricity
- Working local area network connections
- Working Internet connections
- The hardware specified in the Hardware Interfaces Section

### **1.6.3 Constraints**

The System is designed with the following constraints in mind.

The System serves a single library with a single [collection](#). No inter-library loans can be serviced and patrons cannot search for books in other libraries' [collections](#).

## **1.7 Standards Compliance**

The lifetime of the project is expected to be long. It is important for standards to be defined for system design and development.

### 1.7.1 Software Standards

**Methods** The System is to be developed using a formally defined model. The B-Method is used and covers:

1. Analysis, design, and development methods
2. Review procedures
3. Documentation
4. Coding and debugging

For testing, the method mentioned by Sorkin[3] is to be used.

Functional analysis is done using a suitable CASE tool such as Rodin [4].

Detailed design and development standards are not specified, but expected to conform to established practices.

An objected oriented design is encouraged but not required.

**Databases** Collection and Member information are to be kept in a commercial relational database.

### 1.7.2 Hardware Standards

**Barcode Hardware** Any barcode reader used by the System must comply with ISO/IEC 15426-1 standard[5].

**System Hardware** The System should run on consumer grade hardware.

### 1.7.3 Reuse of Existing Software

Whenever possible, the System should leverage existing software. However, all existing software is to be evaluated in terms of the specification document. This helps reduce life-cycle costs and maintenance efforts.

## 2 Functional Requirements

This section details the functional requirements of the System. Requirements that will be discussed include feature descriptions, use cases, user interfaces, hardware interfaces, and software interfaces.

### 2.1 Feature Description

This section contains formal descriptions of functional requirements.

#### 2.1.1 Calculating Book Reservations

Books may be checked out by a library [member](#) for 28 days before they must be renewed. A [member](#) may keep the book for the full 28 days even if a reservation has been placed upon the book. At that time, if a reservation has been placed on the book, it will be due back to the library. The standard processing time for books to be checked back in will be one business day. The system will calculate reservations for a book according to the following specification.

There will be two cases under which book reservations will be conducted. Each of the cases will use the following nomenclature.

Let  $m$  refer to a [member](#) database entry and  $m.reserved[ ]$  be the (zero based) list of reserved books for  $m$ . Let  $c$  refer to a [collection](#) database entry and  $c.Status$  refer to the current status of the book (i.e., checked in, checked out, restricted, reserved, damaged).

Case 1: The book being reserved is not already checked out.

$$\begin{aligned} m.reserved[i].UUID &= c.UUID \\ & \text{if } (c.Status == \text{checked in} \wedge c.Status \neq \text{restricted} \wedge c.Status \neq \text{damaged}) \\ m.reserved[i].date &= NULL \text{ if } (m.reserved[i].UUID \neq NULL) \\ m.reserved[i].number &= 0 \text{ if } (m.reserved[i].UUID \neq NULL) \\ c.Status &= \text{reserved if } (m.reserved[i].UUID \neq NULL) \end{aligned}$$

Case 2: The book being reserved is already checked out.

```

m.reserved[i].UUID = C.UUID
  if(c.Status ≠ checked in ∧ c.Status ≠ restricted ∧ c.Status ≠ damaged)
m.reserved[i].date = c.reservationCount × 30 + c.dueDate
  if (m.reserved[i].UUID ≠ NULL)
m.reserved[i].number = c.reservationCount if (m.reserved[i].UUID ≠ NULL)
c.reservationCount = c.reservationCount + 1 if (m.reserved[i].UUID ≠ NULL)

```

### 2.1.2 Renewing Fees

A midnight every day, the System should check the [member](#) database for all members with checked out books. It then updates fees for each member according to the following specification.

Let *m* refer to a [member](#) database entry with a checked out book and *m.fee* refer to fees for that entry. Let *m.books[ ]* be the (zero based) list of books for *m*. Let *n* be the length of *m.books[ ]*. Let *m.books[i]* refer to a book in the list.

$$m.fee = m.fee + 0.25 * \sum_{i=0}^{n-1} \begin{cases} 1 & \text{if } (m.books[i].DueDate < currentDate) \\ 0 & \text{otherwise} \end{cases}$$

### 2.1.3 Calculating UUIDs

A [UUID](#) is an identifier that is guaranteed to be unique within a set. For most cases it is enough to guarantee that a newly generated UUID be statistically very unlikely to already exist in the given set.

For this software system it is simple to keep track of all of the previously generated UUIDs so we can simply use an incrementing value.

Let *u* be the most recently generated [UUID](#), which is an integer in the range [*a*, *b*]. If no UUID has been generated yet then *u* is defined as *u* = 0. Consequently, no book will ever have a [UUID](#) of 0.

Then a new [UUID](#) can be generated with the following function.

$$UUID(a, b, u) \equiv \{u' | u' = u + 1 \wedge u' \leq b\}$$

## 2.2 Use Cases

This section contains the use cases for the System.

### 2.2.1 Check Out a Book

Actor: [librarian](#)

1. The System prompts the **librarian** to choose an action.
2. The **librarian** selects **Check Out**.
3. The System enters the Member Search screen and prompts the **librarian** to enter a search field.
4. The **librarian** enters the search criteria.
5. The System displays a list of **members** that match the criteria.
6. The **librarian** selects the correct **member** and clicks **Select Member**.
7. The System enters the Book Search screen and prompts the **librarian** to enter a search field.
8. The **librarian** enters the search criteria.
9. The System displays a list of books that match the criteria and are not already checked out.
10. The **librarian** selects the correct book and clicks **Select Book**.
11. A confirmation screen asks the **librarian** to confirm the checkout.
12. The **librarian** selects **Confirm**.
13. Another confirmation screen asks the **librarian** to select **Finished** or **Check Out More Books**.
14. If the **librarian** wants to check out more books, he or she selects **Check Out More Books** and repeat steps 7-12 for each addition book to be checked out.
15. When the **librarian** is done, he or she clicks the **Finished** button.
16. Done-The book(s) are checked out.

### 2.2.2 Check in a Book

Actor: **librarian**

1. The System prompts the **librarian** to choose an action.
2. The **librarian** selects **Check In**.
3. The System enters the Book Search screen and prompts the **librarian** to enter a search field.
4. The **librarian** enters the search criteria.

5. The System displays a list of checked out books that match the criteria.
6. The **librarian** selects the correct book and clicks **Select Book**.
7. A confirmation screen asks the **librarian** to confirm the check in.
8. Done-The book is checked in.

### 2.2.3 Reserve a Book

Actor: **librarian**

1. The System prompts the **librarian** to choose an action.
2. The **librarian** selects **Reserve**.
3. The System enters the Member Search screen and prompts the **librarian** to enter a search field.
4. The **librarian** enters the search criteria.
5. The System displays a list of **members** that match the criteria.
6. The **librarian** selects the correct **member** and clicks **Select Member**.
7. The System enters the Book Search screen and prompts the **librarian** to enter a search field.
8. The **librarian** enters the search criteria.
9. The System displays a list of books that match the criteria.
10. The **librarian** selects the correct book and clicks **Select Member**.
11. A confirmation screen asks the **librarian** to confirm the reservation.
12. The **librarian** selects **Confirm**.
13. Another confirmation screen asks the **librarian** to select **Finished** or **Reserve More Books**.
14. If the **librarian** wants to check out more books, he or she selects **Reserve More Books** and repeat steps 7-12 for each addition book to be reserved.
15. When the **librarian** is done, he or she clicks the **Finished** button.
16. Done-The book(s) are checked out.

### 2.2.4 Book Search

Actor: [patron](#)

1. The [patron](#) navigates to the library's website.
2. The System shows the website including a link to the search.
3. The [patron](#) selects the search link.
4. The System enters the Web Search screen and prompts the user to enter a search field.
5. The [patron](#) enters the appropriate search criteria and clicks **Search**.
6. The System displays a list of matching books.
7. The [patron](#) selects the desired book.
8. The System displays the book's information on the Book Information screen.
9. Done-The book has been searched for.

### 2.2.5 In-Library Member Search

Actor: [librarian](#)

1. The System prompts the [librarian](#) to choose an action.
2. The [librarian](#) enters the [Update Members Interface](#).
3. The System prompts the [librarian](#) to either Add a Member or Search/Update a Member.
4. The [librarian](#) chooses **Search/Update a Member**
5. The System enters the Book Search screen and prompts the [librarian](#) to enter a search field.
6. The [librarian](#) enters the appropriate search criteria and clicks **Search**.
7. The System displays a list of matching [members](#).
8. The [librarian](#) selects the desired [member](#) and clicks **Select Member**.
9. The System displays editable fields for the book and an option to delete the [member](#) from the database.
10. Done-The [librarian](#) has searched for the [member](#).

### 2.2.6 Adding a Book to the Collection

Actor: [librarian](#)

1. The System prompts the [librarian](#) to choose an action.
2. The [librarian](#) enters the [Update Collection Interface](#).
3. The System prompts the [librarian](#) to either Add a Book or Search/Update a Book.
4. The [librarian](#) chooses Add a Book.
5. The System prompts the [librarian](#) to enter the required fields.
6. The [librarian](#) enters the required fields and presses Save Book.
7. Done-The book is now entered into the [collection](#).

### 2.2.7 Updating a Book

Actor: [librarian](#)

1. The System prompts the [librarian](#) to choose an action.
2. The [librarian](#) enters the [Update Collection Interface](#).
3. The System prompts the [librarian](#) to either Add a Book or Search/Update a Book.
4. The [librarian](#) chooses Search/Update a Book
5. The System enters the Book Search screen and prompts the [librarian](#) to enter a search field.
6. The [librarian](#) enters the appropriate search criteria and clicks Search.
7. The System displays a list of matching books.
8. The [librarian](#) selects the desired book and clicks Select Book.
9. The System displays editable fields for the book and an option to delete the book from the [collection](#).
10. The [librarian](#) fills out the required fields and clicks Save Book.
11. Done-The book is updated.



### 2.2.8 Deleting a Book from the Collection

Actor: [librarian](#)

1. The System prompts the [librarian](#) to choose an action.
2. The [librarian](#) enters the [Update Collection Interface](#).
3. The System prompts the [librarian](#) to either Add a Book or Search/Update a Book.
4. The [librarian](#) chooses Search/Update a Book
5. The System enters the Book Search screen and prompts the [librarian](#) to enter a search field.
6. The [librarian](#) enters the appropriate search criteria and clicks Search.
7. The System displays a list of matching books.
8. The [librarian](#) selects the desired book and clicks Select Book.
9. The System displays editable fields for the book and an option to delete the book from the [collection](#).
10. The [librarian](#) Delete Book.
11. Done-The book is deleted.

### 2.2.9 Adding a Member

Actor: [librarian](#)

1. The System prompts the [librarian](#) to choose an action.
2. The [librarian](#) enters the [Update Members Interface](#).
3. The System prompts the [librarian](#) to either Add a Member or Search/Update a Member.
4. The [librarian](#) chooses Add a Member.
5. The System prompts the [librarian](#) to enter the required fields.
6. The [librarian](#) enters the required fields and clicks Save Member.
7. Done-The member is now entered into the database.

### 2.2.10 Updating a Member

Actor: librarian

1. The System prompts the librarian to choose an action.
2. The librarian enters the Update Members Interface.
3. The System prompts the librarian to either Add a Member or Search/Update a Member.
4. The librarian chooses Search/Update a Member.
5. The System enters the Member Search screen and prompts the librarian to enter a search field.
6. The librarian enters the appropriate search criteria and clicks Search.
7. The System displays a list of matching members.
8. The librarian selects the desired member and clicks Select Member.
9. The System displays editable fields for the member and an option to delete the member from the database.
10. The librarian fills out the required fields and clicks Save Member.
11. Done-The member is updated.

### 2.2.11 Deleting a Member

Actor: librarian

1. The System prompts the librarian to choose an action.
2. The librarian enters the Update Members Interface.
3. The System prompts the librarian to either Add a Member or Search/Update a Member.
4. The librarian chooses Search/Update a Member.
5. The System enters the Member Search screen and prompts the librarian to enter a search field.
6. The librarian enters the appropriate search criteria and clicks Search.
7. The System displays a list of matching members.
8. The librarian selects the desired member and clicks Select Member.

9. The System displays editable fields for the **member** and an option to delete the **member** from the database.
10. The **librarian** clicks **Delete Member**.
11. Done-The **member** is deleted.

#### **2.2.12 Pay Fees**

Actor: **librarian**

1. The System prompts the **librarian** to choose an action.
2. The **librarian** enters the **Update Members Interface**.
3. The System prompts the **librarian** to either **Add a Member** or **Search/Update a Member**.
4. The System prompts the **librarian** to enter search criteria.
5. The **librarian** enters the appropriate search criteria and clicks **Search**.
6. The System displays a list of matching **members**.
7. The **librarian** selects the desired **member** and clicks **Select Member**.
8. The System displays editable fields for the **member** and an option to delete the **member** from the database.
9. The **librarian** enters the fee amount the **member** is paying and clicks **Save Member**.
10. Done-The fees are paid.

#### **2.2.13 Renew Book**

Actor: **librarian**

1. The System prompts the **librarian** to choose an action.
2. The **librarian** selects **Renew**.
3. The System enters the Book Search screen and prompts the **librarian** to scan the barcode or enter a search field.
4. The **librarian** enters the search criteria and clicks **Search**.
5. The System displays a list of books that match the criteria.
6. The **librarian** selects the correct book and clicks **Select Book**.
7. A confirmation screen asks the **librarian** to confirm the renewal.
8. Done-The book is renewed.

### 2.2.14 Failed Check Out

Actor: [librarian](#)

1. The System prompts the [librarian](#) to choose an action.
2. The [librarian](#) selects **Check Out**.
3. The System enters the Member Search screen and prompts the [librarian](#) to enter a search field.
4. The [librarian](#) enters the search criteria.
5. The System displays a list of [members](#) that match the criteria and are not already checked out.
6. The [librarian](#) selects the correct [member](#) and clicks **Select Member**.
7. The System enters the Book Search screen and prompts the [librarian](#) to enter a search field.
8. The [librarian](#) enters the search criteria.
9. The System displays a list of books that match the criteria and are not already checked out.
10. The [librarian](#) selects the correct book and clicks **Select Book**.
11. The System informs the [librarian](#) using a Confirmation screen that the book is either reserved to another [member](#) or that [member](#) has an outstanding fee.
12. Done-The book has failed to be checked out.

### 2.2.15 Failed Search

Actor: [patron](#)

1. The [patron](#) navigates to the library's website.
2. The System shows the website including a link to the search.
3. The [patron](#) selects the search link.
4. The System enters the Web Search screen and prompts the [patron](#) to enter a search field.
5. The [patron](#) enters the appropriate search criteria and clicks **Search**.
6. No books have matched the search, the System informs the [patron](#) that there are no matching books. The System allows the [patron](#) to re-enter the search fields.
7. Done-The book search has failed.

### 2.2.16 Failed In-Library Member Search

Actor: [librarian](#)

1. The System prompts the [librarian](#) to choose an action.
2. The [librarian](#) enters the [Update Members Interface](#).
3. The System prompts the [librarian](#) to either Add a Member or Search/Update a Member.
4. The [librarian](#) chooses Search/Update a Member
5. The System enters the Book Search screen and prompts the [librarian](#) to enter a search field.
6. The [librarian](#) enters the appropriate search criteria and clicks Search.
7. No [members](#) have matched the search, the System informs the [librarian](#) that there are no matching [members](#). The System allows the [librarian](#) to re-enter the search fields.
8. Done-The [member](#) search has failed.

### 2.2.17 Failed Adding a Book to the Collection

Actor: [librarian](#)

1. The System prompts the [librarian](#) to choose an action.
2. The [librarian](#) enters the [Update Collection Interface](#).
3. The System prompts the [librarian](#) to either Add a Book or Search/Update a Book.
4. The [librarian](#) chooses Add a Book.
5. The System prompts the [librarian](#) to enter the required fields.
6. The [librarian](#) enters the required fields and presses Save Book.
7. One or more of the fields contains invalid input. The System informs the [librarian](#) of the situation and the [librarian](#) is allowed to re-enter data.
8. Done-Adding a book has failed.

### 2.2.18 Failed Updating a Book

Actor: [librarian](#)

1. The System prompts the [librarian](#) to choose an action.
2. The [librarian](#) enters the [Update Collection Interface](#).
3. The System prompts the [librarian](#) to either Add a Book or Search/Update a Book.
4. The [librarian](#) chooses Search/Update a Book
5. The System enters the Book Search screen and prompts the [librarian](#) to scan the barcode or enter a search field.
6. The [librarian](#) enters the appropriate search criteria and clicks Search.
7. The System displays a list of matching books.
8. The [librarian](#) selects the desired book and clicks Select Book.
9. The System displays editable fields for the book and an option to delete the book from the [collection](#).
10. The [librarian](#) fills out the required fields and clicks Save Book.
11. One or more of the fields contains invalid input. The System informs the [librarian](#) of the situation and the [librarian](#) is allowed to re-enter data.
12. Done-The book has failed to be updated.

### 2.2.19 Failed Adding a Member

Actor: [librarian](#)

1. The System prompts the [librarian](#) to choose an action.
2. The [librarian](#) enters the [Update Members Interface](#).
3. The System prompts the [librarian](#) to either Add a [Member](#) or Search/Update for a Member.
4. The [librarian](#) selects Add a [Member](#).
5. The System prompts the [librarian](#) to enter the required fields.
6. The [librarian](#) enters the required fields and clicks Search.
7. One or more of the fields contains invalid input. The System informs the [librarian](#) of the situation and the [librarian](#) is allowed to re-enter data.
8. Done-The member has failed to be entered into the database.

### 2.2.20 Failed Updating a Member

Actor: **librarian**

1. The System prompts the **librarian** to choose an action.
2. The **librarian** enters the **Update Members Interface**.
3. The System prompts the **librarian** to either Add a Member or Search/Update a Member.
4. The **librarian** chooses **Search/Update a Member**.
5. The System enters the Member Search screen and prompts the **librarian** to enter a search field.
6. The **librarian** enters the appropriate search criteria and clicks **Search**.
7. The System displays a list of matching **members**.
8. The **librarian** selects the desired **member** and clicks **Select Member**.
9. The System displays editable fields for the **member** and an option to delete the **member** from the database.
10. The **librarian** fills out the required fields and clicks **Save Member**.
11. One or more of the fields contains invalid input. The System informs the **librarian** of the situation and the **librarian** is allowed to re-enter data.
12. Done-The **member** has failed to be updated.

### 2.2.21 Failed Deleting a Member

Actor: **librarian**

1. The System prompts the **librarian** to choose an action.
2. The **librarian** enters the **Update Members Interface**.
3. The System prompts the **librarian** to either Add a Member or Search/Update a Member.
4. The **librarian** chooses **Search/Update a Member**.
5. The System enters the Member Search screen and prompts the **librarian** to enter a search field.
6. The **librarian** enters the appropriate search criteria and clicks **Search**.

7. The System displays a list of matching **members**.
8. The **librarian** selects the desired **member** and clicks **Select Member**.
9. The System displays editable fields for the **member** and an option to delete the **member** from the database.
10. The **librarian** clicks **Delete Member**.
11. The member currently has an unpaid fee. The System informs the **librarian** of the situation.
12. Done-The **member** has failed to be deleted.

### 2.2.22 Cancel Action

This is a general use case that can apply to any action that takes the **librarian** away from the Primary screen. The point is the librarian can cancel what he or she is doing at any time.

Actor: **librarian**

1. The System prompts the **librarian** to choose an action.
2. The **librarian** starts to complete an action as illustrated by the other use cases.
3. The **librarian** decided that he or she does not need to complete the action and presses **Cancel**.
4. Done-The action has been cancelled.

## 2.3 User Interfaces

The System makes use of two separate user interfaces. The **Library Interface** is used to manage books and **members**. It consists of eight types of screens and is only meant to be used in the library. The **Web Interface** is accessed online via a Web browser and consists of two screens.

### 2.3.1 Library Interface

The Primary screen (figure 1) for the **Library Interface** consists of six large buttons. They are **Check Out**, **Check In**, **Renew**, **Reserve**, **Update Members**, and **Update Collection**. The buttons all need to be large to ensure they can be quickly accessed. Checking in, checking out, reserving, and renewing are all actions that need to be done quickly and will be done often. Therefore, they each get their own button.

The **Update Collection Interface** has a primary screen (figure 2) with three buttons. They are **Add a Book**, **Search/Update a Book**, and **Cancel**.



The [Update Members Interface](#) has a primary screen (figure 3) with three buttons. They are `Add a Member`, `Search/Update a Member`, and `Cancel`.

The Book Search screen (figure 4) includes a keyword search box (for general searches), a search box for each of the searchable fields described in the feature descriptions section, a `Search` button, a `Cancel` button, and a `Select Book` button. Once the [librarian](#) clicks `Search`, a list of books that match the criteria will appear under the search fields.

The Member Search screen (figure 5) includes a keyword search box (for general searches), a search box for each of the searchable fields described in the feature descriptions section, a `Search` button, a `Cancel` button, and a `Select Member` button. Once the [librarian](#) clicks `Search`, a list of [members](#) that match the criteria will appear under the search fields.

The Edit Book screen (figure 6) contains edit text boxes that enable the [librarian](#) to change any field associated with a book other than the `UUID`. It also has a `Delete Book` button, a `Save Book` button, and a `Cancel` button.

The Edit Member screen (figure 7) contains edit text boxes that enable the [librarian](#) to change any field associated with a [member](#) other than the `UUID`. In addition, it contains a list of checked out books and a list of reserved books with an `Unreserve` button for each book. It has a `Delete Member` button, a `Save Member` button, and a `Cancel` button. It also has a fees field for Fees that the [librarian](#) can use to take payment for unpaid fees.

The Confirmation screen (figure 8) comes in several varieties. It serves to verify an action, ask the [librarian](#) to confirm an action, or inform the [librarian](#) that an action cannot be completed.

### 2.3.2 Web Interface

The Web Search screen (figure 9) includes a keyword search box (for general searches), a search box for each of the searchable fields described in the Feature Description section, an extra field representing the status of each book, and a `Search` button. Once the user clicks `Search`, a list of books that match the criteria will appear under the search fields. Each result in the Web Search screen is a hyperlink that takes the [patron](#) to the Book Information screen for that book. This interface was inspired by the search interface at Albertsons Library [6].

The Book Information screen (figure 10) lists all of the fields described in the Feature Description section of this document.

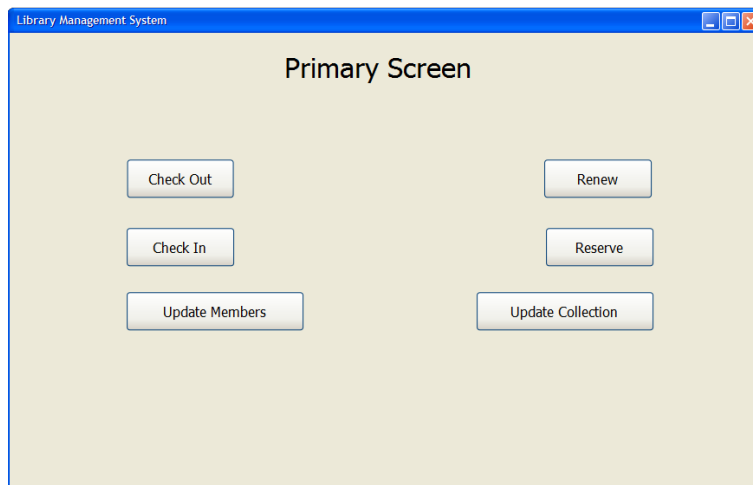


Figure 1: The Primary screen.

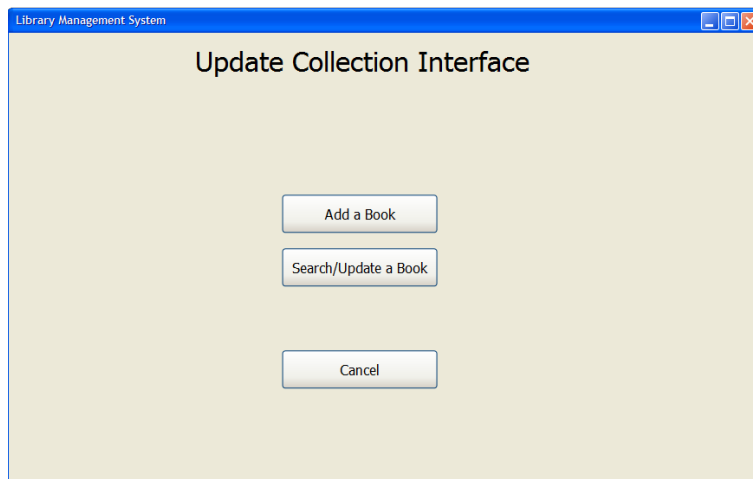


Figure 2: The first screen of the [Update Collection Interface](#).

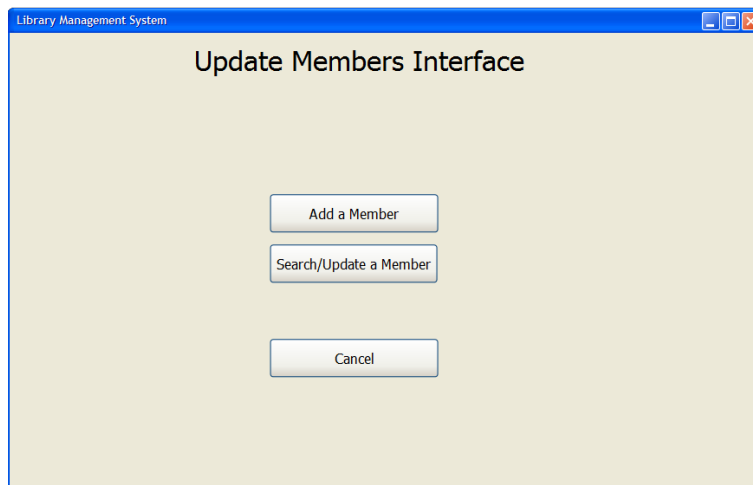


Figure 3: The first screen of the [Update Members Interface](#).

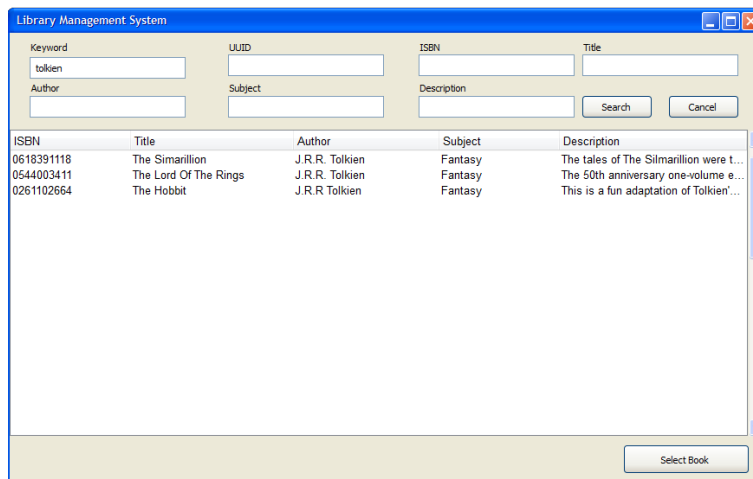


Figure 4: The Book Search screen.

UUID	First Name	Last Name	Mailing Address	Phone Number
122333	Bob	Smith	123 Fake Street Boise, ID, 83705	208-555-5555
12321	Bob	Johnson	5678 Made Up Lane, La Grande, OR, 97850	541-555-5555
456323	Bob	Bobson	9 Nonexistant Row, Boise, ID, 83705	208-555-5556

Figure 5: The Member Search screen.

**Edit Book**

UUID: 1234      Call Number: JFIC TOLKIEN J.      ISBN: 0261102664

Title: The Hobbit      Subject: Fantasy      Author: J.R.R. Tolkien

Edition: First Thus edition      Year: 1998      Status: Checked In

Type: Fiction      Due Date:      Description: This is a fun adaptation of Tolkien's

Reservation Count: 1

Buttons: Delete Book, Cancel, Save Book

Figure 6: The Edit Book screen.

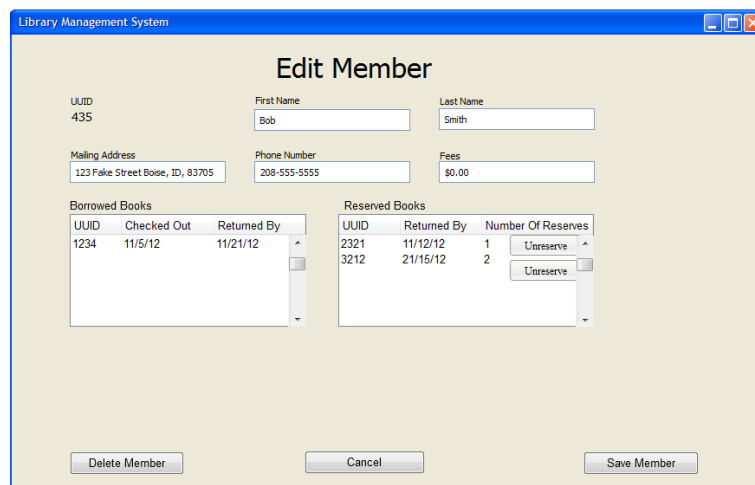


Figure 7: The Edit Member screen.

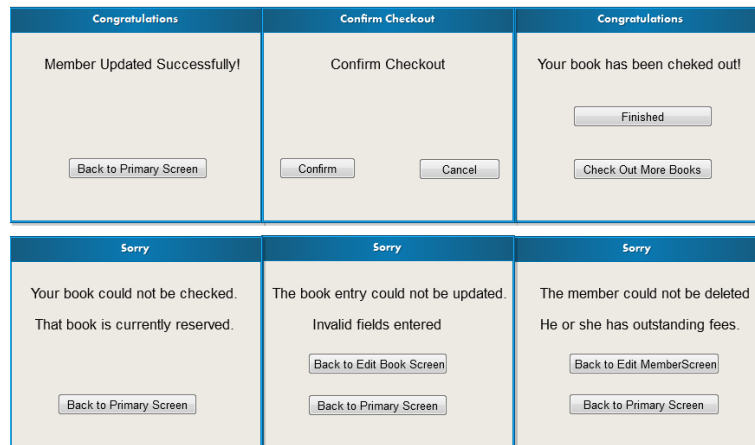


Figure 8: Several example Confirmation screens.

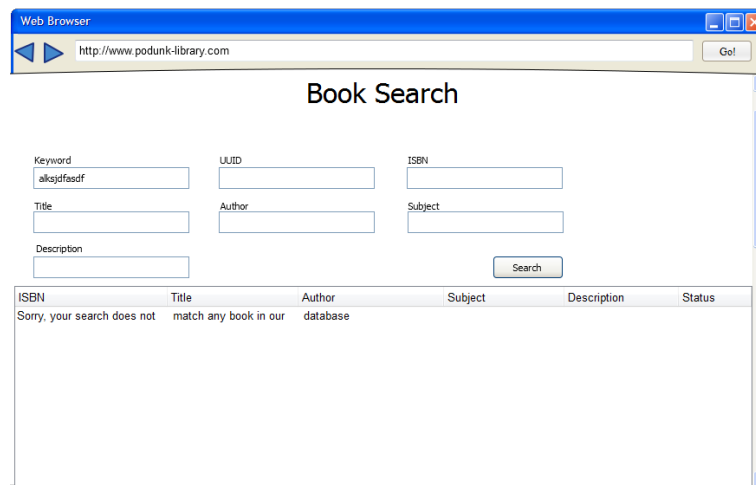


Figure 9: The Web Search screen.

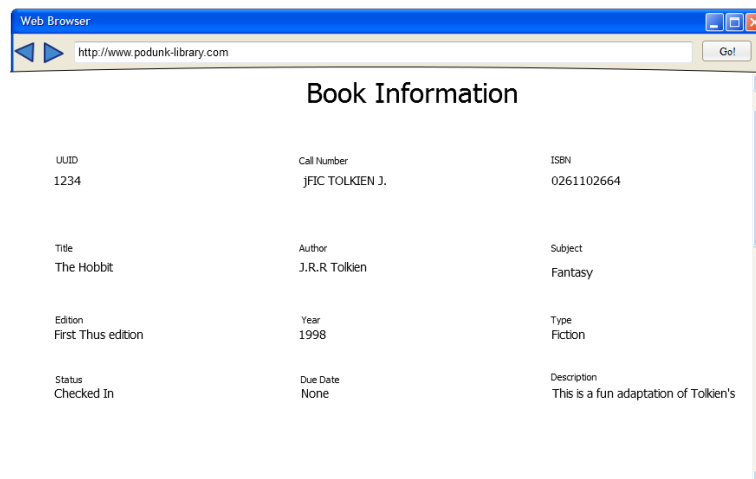


Figure 10: The Book Information screen.

## 2.4 Hardware Interfaces

The [Library Interface](#) requires a desktop computer (including a tower, monitor, keyboard, and mouse).

In order to expedite the process of entering a book or [member](#), the hardware interface must also include a barcode scanner and a printer capable of printing labels for books and cards for [members](#).

Any computer with a browser and internet access can access the [Web Interface](#).

The background database and Web server must be hosted on an appropriate database and Web server.

## 2.5 Software Interfaces

The software for the [Library Interface](#) must interact with a background database in order to maintain the [collection](#) of books and the library's [members](#).

The [Web Interface](#) must interact with a web server and the background database.

## 3 Nonfunctional Requirements

The following are the nonfunctional requirements for the System.

### 3.1 Time and Space Performance

This section specifies the speed requirements of the System.

**Library Interface** The **Library Interface** itself will be able to perform searches of the database in **constant time**. This applies to searches of the **collection** and **member** databases. Book checkout will also be performed in **constant time**. This only requires the **member**'s account and the **collection** be updated to reflect that the book is no longer available.

**Web Interface** The **Web Interface** of the System will be able to fetch **patron** requested data from the database in **constant time**. This means that the **patron** will quickly receive the search that they requested, or a not found message. The end performance as perceived by the **patron** will be affected by factors other than search performance. Such factors include personal computer performance and Internet speed, which are not under the control of the System.

### 3.2 Reliability

This section specifies the required reliability of both the **Library Interface** and the **Web Interface**.

**Library Interface** The reliability of the **Library Interface** will depend on two factors. The first factor is the reliability of the library's computer systems. This is out of the control of the System, which can only perform as well as the computer that it is used on. When the System is used on a reliable and well suited device, the software should be no less than 99% reliable. This reliability statistic is determined by the amount of downtime that a typical library will experience with the System. This System will perform checks on all of the data that a librarian enters into any field, and will ensure that it is a valid and logical entry for that field. These checks will reduce the problems that the System or database could experience with bad data inputs.

The second factor, is the reliability of the database for the System. To ensure the 99% reliability rating, the database will be made resilient and reliable in two ways. First, all of the input data that it is being passed will be checked for validity by the main program, thus reducing problems associated with bad data. Second, the database will perform self backups on a regular schedule. This schedule will be determined by the



System. The System will determine the need for backups or disc checks of the database based on:

1. A count of past system failures.
  - Each time the System is shut down incorrectly the System will increment a failure count on the next start up.
2. System load history
  - In a library with a very low load the System will perform disc checks on a timed schedule.
  - In a library with a larger load the System will perform disc checks after each full transaction (pending other backup factors).
3. Number of terminals running the System
  - With more terminals running the disc checks will occur more frequently.

**Web Interface** The reliability of the **Web Interface** will be dependent on two factors. The first is the reliability of the server that is hosting the library site. The reliability of the server is out of the control of the System and this specification. Second, the **Web Interface** reliability will be determined by the reliability of the **Library Interface** database. When the System is up and running, the **Web Interface** will be fully functional.

### 3.3 Portability

The System will have the following portability features.

**Library Interface** The **Library Interface** of the System will be portable to different operating systems. A version will be available for both Linux and Windows. This will ensure that different libraries will be able to implement the System. This portability will also be useful in libraries that use multiple operating systems for daily use. This will ensure that the library will not need a single computer dedicated to the servicing of library patrons. The library can use the interface on any number of computers in a manner that is best suited to the individual library's layout and needs.

**Web Interface** The **Web Interface** for the System is portable by definition. The System will adhere to common conventions of Web programming in its implementation and maintenance. This will ensure that the search functionality will be maintained across different Web browsers.

### **3.4 Security**

The security of the System will rely on the System set-up and use. It is recommended that the computer system on which the [Library Interface](#) is installed be password protected upon start-up and when left unattended. This will prevent unauthorized access to the System.

The second feature of the System that needs to be secured is the database. This will be achieved by securing the database from common Web attacks. This ensures the library's books, employees, and patrons remain safe.

## 4 System Tests

This section details the estimated number of tests for each part of the System. These include the front-end system used by [librarians](#), the [Web Interfaces](#) used by patrons in the library, and the database that holds the entire system.

### 4.1 Test Time and Coverage Estimate

To estimate the number of tests and time that will be required to confirm the working order of each of the components of the System, the methodology as suggested by Sorkin [3] will be used. Sorkin suggests that a software system be broken down into components, and that each component be given a risk value of “High”, “Med”, or “Low”. With each of the components ranked, a number of tests can be assigned based on that rating. Using his system more tests are assigned to the higher risk components. Below is a table detailing the components of the System, the anticipated risk value of each component, and the minimum number of tests that are expected to be needed to confirm functionality.

	Risk	Est. Tests
Screens:		
Primary Screen	Low	5
Update Collection Interface	Low	7
Update Members Interface	Low	7
Book Search Screen	Med	10
Member Search Screen	Med	10
Edit Book Screen	High	25
Edit Member Screen	High	25
Confirmation Screen	Low	2
Web Search Screen	Med	15
Database		
Database Security	High	25+
Database I/O	High	25+
Total		156+

Test Planning - 15/day, 156+/15 = 11 days

Test Execution - 25/day, 156+/25 = 7 days

Debug & Fix - Retests @50 = 2 days

TOTAL = 20 days

## Glossary

**collection** The set of books that the library owns for the purpose of loaning to library members. [iv](#), [1–5](#), [8–11](#), [13](#), [18](#), [19](#), [23](#), [24](#), [33](#), [34](#)

**constant time** An algorithm is said to be constant time (also written as  $O(1)$  time) if the value of  $T(n)$  is bounded by a value that does not depend on the size of the input. [34](#)

**damaged** One of several possible states a book can be assigned. A book is in the damaged state if it has sustained physical damage such as torn pages, broken bindings, or permanent ink markings. [3](#)

**ISBN** A numbering system which uniquely identifies books for commercial purposes. [3](#), [5](#), [6](#)

**LCC** Library of Congress Classification. Also known as a Library of Congress Call Number. This classification identifies a particular book and serves as an address that can be used to locate a copy of the book within the library. [3](#), [6](#)

**librarian** A librarian is an employee or volunteer in the library. [1](#), [2](#), [4–11](#), [14–27](#), [37](#)

**Library Interface** The part of the System that is only accessible to librarians. [iv](#), [1](#), [2](#), [26](#), [33–36](#)

**loan period** A period of 28 days. Members can borrow a book from the library for an amount of time not exceeding the loan period. [7](#)

**member** A person who is registered in the member database for the purpose of borrowing books from the library. [iv](#), [1](#), [2](#), [4–10](#), [13–17](#), [19–27](#), [33](#), [34](#)

**patron** Any person who uses the Web interface to search the collection. [1](#), [6](#), [8](#), [9](#), [17](#), [22](#), [27](#), [34](#)

**relevance** The likelihood that a particular book is the one a patron is searching for. [6](#)

**reserved** One of several possible states a book can be assigned. A book is in this state if a member has placed a hold on the book. A book in this state may not be checked out by any member except the member who has the oldest reservation. [3](#)

**restricted** One of several possible states a book can be assigned. A book is restricted if for some reason it cannot be taken out of the library. Some books such as reference materials are always restricted. [3](#)

**Update Collection Interface** A user interface within the System which allows a librarian to update the information in the collection. [vi](#), [2](#), [18](#), [19](#), [23](#), [24](#), [26](#), [28](#)

**Update Members Interface** A user interface within the System which allows a librarian to update member information or search for members. [vi](#), [2](#), [17](#), [19–21](#), [23–25](#), [27](#), [29](#)

**UUID** An identifier that is guaranteed to be unique among all of the identifiers in a set. [3–5](#), [8](#), [14](#), [27](#)

**Web Interface** The online search system. Usable by anyone with access to the library's Website to search for a book. [2](#), [26](#), [33–35](#), [37](#)

## Appendix A

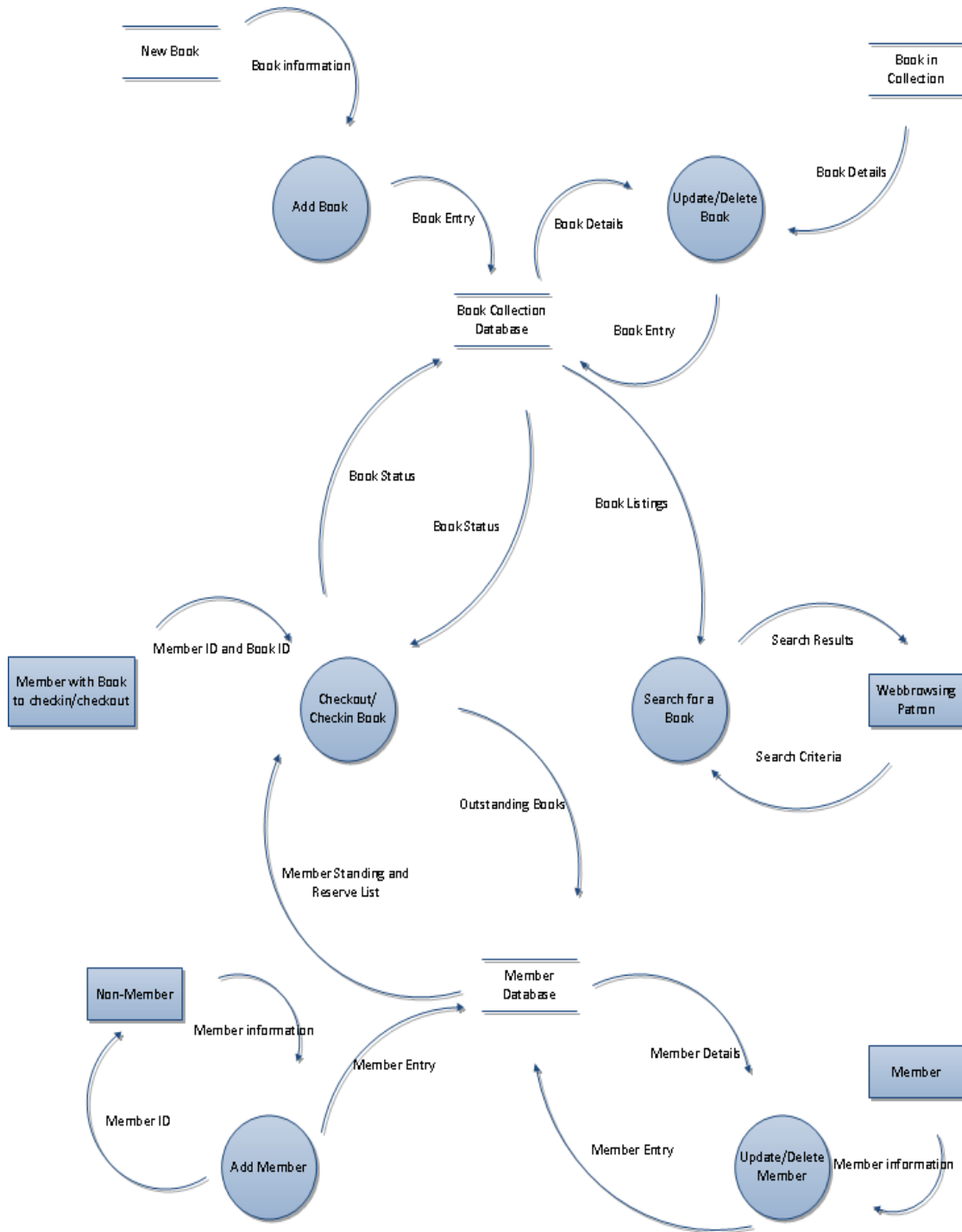


Figure 11: Dataflow Diagram

## References

- [1] Library of congress classification outline. <http://www.loc.gov/catdir/cpsolcco/>, November 2012. Retrieved from Library of Congress website. 1
- [2] LibLime. Liblime koha. <http://www.koha.org/>, November 2012. An open source library management system. 1
- [3] Samford Sorkin. System testing without a specification. <http://costaltech.com/nospec.pdf>, November 2012. 12, 37
- [4] M. Butler and S. Hallerstede. The rodin formal modelling tool. In BCS-FACS Christmas 2007 Meeting-Formal Methods In Industry, London., 2007. 12
- [5] Information technology – Automatic identification and data capture techniques, 2006. 12
- [6] Boise State University. Albertsons library. <http://library.boisestate.edu/>, November 2012. An example library website. 27