Requirements Engineering

Library System

Submitted By: Jason O Connor (T00227173)

Computing with Software Development

Date Submitted: 18/04/2024

Table of Contents

[1 Introduction/Overview 4](#_Toc164625340)

[2 Functional Components 5](#_Toc164625341)

[3 User Requirements 6](#_Toc164625342)

[3.1 LibrarySYS will manage Members 6](#_Toc164625343)

[3.1.1 LibrarySYS will allow a member to be added. 6](#_Toc164625344)

[3.1.2 LibrarySYS will allow a member to view details. 6](#_Toc164625345)

[3.1.3 LibrarySYS will allow a member to be removed. 6](#_Toc164625346)

[3.1.4 LibrarySYS will allow a members details to be updated. 6](#_Toc164625347)

[3.1.5 LibrarySYS will allow a member to pay their fees. 6](#_Toc164625348)

[3.2 LibrarySYS will manage Books Inventory 6](#_Toc164625349)

[3.2.1 LibrarySYS will catalogue a book. 6](#_Toc164625350)

[3.2.2 LibrarySYS will allow a member to search for a book. 6](#_Toc164625351)

[3.3 LibrarySYS will process Loans 7](#_Toc164625352)

[3.3.1 LibrarySYS will record an issued loan. 7](#_Toc164625353)

[3.3.2 LibrarySYS will record a book return. 7](#_Toc164625354)

[3.4 LibrarySYS will allow Admin to review book loans 7](#_Toc164625355)

[3.4.1 LibrarySYS will Review genre popularity. 7](#_Toc164625356)

[4 System Requirements 7](#_Toc164625357)

[4.1 System Level Use Case Diagram 8](#_Toc164625358)

[4.2 Manage Members 9](#_Toc164625359)

[4.2.1 Register Member 9](#_Toc164625360)

[4.2.2 Remove Member 12](#_Toc164625361)

[4.2.3 View Member Details 14](#_Toc164625362)

[4.2.4 Update Member 16](#_Toc164625363)

[4.2.5 Pay Fee 18](#_Toc164625364)

[4.3 Books 20](#_Toc164625365)

[4.3.1 Catalogue Book 21](#_Toc164625366)

[4.3.2 Search Catalogue 22](#_Toc164625367)

[4.4 Loans 24](#_Toc164625368)

[4.4.1 Record Loan 25](#_Toc164625369)

[4.4.2 View Loan 26](#_Toc164625370)

[4.4.3 Return Book 27](#_Toc164625371)

[4.5 Admin 30](#_Toc164625372)

[4.5.1 Review Loans 30](#_Toc164625373)

[5 System Model 32](#_Toc164625374)

[5.1 Level-0 DFD 32](#_Toc164625375)

[5.2 Level-1 DFD 33](#_Toc164625376)

[5.3 Level-2 DFD(Process P1: Members) 34](#_Toc164625377)

[5.4 Level-2 DFD(Process P2: Books) 35](#_Toc164625378)

[5.5 Level-2 DFD(Process P3: Loans) 36](#_Toc164625379)

[5.6 Level-2 DFD(Process P4: Admin) 37](#_Toc164625380)

[6 Data Model 38](#_Toc164625381)

[6.1 Class Diagram 39](#_Toc164625382)

[6.2 Relational Schema 40](#_Toc164625383)

[6.3 Database Schema 41](#_Toc164625384)

[7 Conclusion 43](#_Toc164625385)

# Introduction/Overview

This is a requirements document that will detail the specifications and requirements of a library system. The system will have more focus on user functionality than on administrative ability. The system will have 11 functions.

# Functional Components

# User Requirements

## LibrarySYS will manage Members

### LibrarySYS will allow a member to be added.

### LibrarySYS will allow a member to view details.

### LibrarySYS will allow a member to be removed.

### LibrarySYS will allow a members details to be updated.

### LibrarySYS will allow a member to pay their fees.

## LibrarySYS will manage Books Inventory

### LibrarySYS will catalogue a book.

### LibrarySYS will allow a member to search for a book.

## LibrarySYS will process Loans

### LibrarySYS will record an issued loan.

### LibrarySYS will record a book return.

## LibrarySYS will allow Admin to review book loans

### LibrarySYS will Review genre popularity.

# System Requirements

Member Requirements:

* Librarian will add a member’s information to the Members table of the system.
* Librarian and Members can view member details.
* Librarian and Members can Update Members information in the Members Table.
* Librarian can Remove a Member’s Information from the Members Table.

Books Requirements:

* Member can request to loan book/s.
* Member can search for available book/s in the Books Table.
* Librarian can loan book/s to Member.
* Member can return Book.

Admin Requirements:

* Librarian can review issued Loans to compare genre popularity.

## System Level Use Case Diagram

The following system level use case diagram illustrates the high-level system requirements.

Librarian

Member

## Manage Members

This Module will have multiple functions such as the ability to add and remove a member, view member details, update a member’s information, and allow a member to pay fees.

### Register Member

This function will be used to add a new Member’s data to the Members table in the database. There will also be a “Strike Count” initialised as 0, which will record the number of late returns a member has had, and a fee record that is applied upon a late return. It will also give a Member status to a member to state whether the member is active or has had their account deactivated. It will be an interaction between the System and the Member.



Member

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Register Member | |
| **Use Case Id** | 1 | |
| **Priority** | 1 | |
| **Source** | Member | |
| **Primary Business Actor** | Member | |
| **Other Participating Actors** |  | |
| **Description** | This function is used to save a new Member’s data to the Members file. | |
| **Preconditions** |  | |
| **Trigger** | A new Member would like to be registered. | |
| **Expected Scenario** | **Customer** | **System** |
|  | **Step 1:** Start Register Member function.  **Step 3:** Enter Member Information.   * Surname * First name * Address   + House Number   + Street   + Town   + County   + Eircode * Phone * Email     **Step 4:** Confirm Registration.                          **Step 9:** Confirm OK. | **Step 2:** Display UI.    **Step 5:** Validate data submitted, All fields must have entries:   * Surname(varchar2) , must not be numeric. * Forename(varchar2) , must not be numeric. * Address(   House Number(varchar2)  Street(varchar2)  Town(varchar2)  County(varchar2)  Eircode(varchar2)  )   * Phone(numeric,                Number must be 10 digits)   * Email(Varchar2)   **Step 6:** Generates next available Member ID by incrementing the greatest member id by 1.  **Step 7:** Saves new Member information in the Members file:   * MemberID(numeric) * Surname (Varchar2) * Forename (Varchar2) * Address(   House Number (Varchar2)  Street (Varchar2)  Town (Varchar2)  County (Varchar2)  Eircode (Varchar2)  )   * Phone(numeric) * Email(Varchar2) * Fee Amount(numeric, default = 0) * Strike Count(numeric, default = 0) * Status(Varchar1, default = “A”)   **Step 8:** Display Confirmation message.  **Step 10:** Reset UI. |
| **Alternate Scenario** | **Customer** | **System** |
| **Invalid data** |  | **Step 5:** Validation failed.  **Step 6:** Display appropriate error message.  **Step 7:** Return to Step 3. |
| **Conclusions** | A new Member has been added to the Members file in the database. | |
| **Post conditions** | The Member can now borrow books. | |
| **Business Rules** |  | |

### Remove Member

This function will be used to Remove a Member’s data from the Members table in the database. It will check if the member has a strike count equal or greater than 3, and then it will change the member status to ‘D’. It will be an interaction between the Librarian and the System.



Librarian

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Remove Member | |
| **Use Case Id** | 2 | |
| **Priority** | 1 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** |  | |
| **Description** | This function is used to remove an existing Member’s data from the Members file. | |
| **Preconditions** |  | |
| **Trigger** | A Member has received 3 strikes and will have its membership deactivated. | |
| **Expected Scenario** | **Customer** | **System** |
|  | **Step 1:** Start Remove Member function.  **Step 3:** Enter Member Information.   * MemberID   **Step 7:** Confirm Removal. | **Step 2:** Display UI.    **Step 4:** Validate entered data:  MemberID must be entered.  **Step 5:** Retrieve Member details from Members file and display on UI.   * Surname (Varchar2) * Forename (Varchar2) * Address(   House Number (Varchar2)  Street (Varchar2)  Town (Varchar2)  County (Varchar2)  Eircode (Varchar2)  )   * Phone(numeric) * Email(Varchar2) * Fee Amount(numeric, default = 0)   **Step 6:** Check StrikeCount >= 3 in the Members file.  **Step 8:** Display Confirmation message.    **Step 9:** Change Member Status to “D” for deactivated.  Where (MemberID = **Step 3** MemberID)  **Step 10:** Display Update message:  “Member -MemberID- has been removed”  **Step 11:** Reset UI. |
| **Alternate Scenario** | **Customer** | **System** |
| **Invalid MemberID** |  | **Step 4:** Validation failed.  **Step 5:** Display appropriate error message.  **Step 6:** Return to Step 3. |
| **Conclusions** | A Member has been Removed from the Members file in the database. | |
| **Post conditions** | The Member can no longer be loaned books from the Library. | |
| **Business Rules** |  | |

### View Member Details

This function will allow a member to view their information. It also allows a member to transfer to the pay fees function if their due fees are > 0. It is an interaction between the Member and the System.



Member

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | View Member Details | |
| **Use Case Id** | 3 | |
| **Priority** | 2 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Member | |
| **Other Participating Actors** |  | |
| **Description** | This function will be used to View a Member’s details, allowing a member to check fees owed and strike count. | |
| **Preconditions** | The Member must have a valid MemberID. | |
| **Trigger** | A Member would like to view their information. | |
| **Expected Scenario** | Member | **System** |
|  | **Step 1:** Start View Member Details function.  **Step 3:** Enter Member Information.   * MemberID | **Step 2:** Display UI.    **Step 4:** Validate entered data:  Check that the entered data has an entry in the Members file in the database by matching MemberID.  **Step 5:**  Retrieve member details from the members file and display on UI.   * Surname (Varchar2) * Forename (Varchar2) * Address(   House Number (Varchar2)  Street (Varchar2)  Town (Varchar2)  County (Varchar2)  Eircode (Varchar2)  )   * Phone(numeric) * Email(Varchar2) * Fee Amount(numeric, default = 0) |
| **Alternate Scenario** | **Member** | **System** |
| **Invalid MemberID** |  | **Step 5:** Validation failed.  **Step 6:** Display appropriate error message.  **Step 7:** Return to Step 4. |
| **Fee Amount > 0** | **Step 7:** Confirm Pay Fee. | **Step 6:** Display Pay Fee Button.  **Step 8:** Load Pay Fee function. |
| **Conclusions** | A Member has Updated information into the Members file. | |
| **Post conditions** | The Member’s information can be updated as long as they have a valid MemberID. | |
| **Business Rules** | Must have a Valid MemberID. | |

### Update Member

This function will Update a Member’s data in the Members file, allowing a member to change their details post-registration. It is an interaction between the Member and the System.



Member

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Member | |
| **Use Case Id** | 4 | |
| **Priority** | 2 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Member | |
| **Other Participating Actors** |  | |
| **Description** | This function will be used to Update Member’s data in the Members table in the database. | |
| **Preconditions** | The Member must have a valid MemberID. | |
| **Trigger** | A Member would like to update their information. | |
| **Expected Scenario** | Member | **System** |
|  | **Step 1:** Start Update Member information function.  **Step 3:** Enter Member Information.   * MemberID         **Step 6:** Enter new information in the required fields and submit. | **Step 2:** Display UI.    **Step 4:** Validate entered data:  Check that the entered data has an entry in the Members file in the database by matching MemberID.  **Step 5:**  Retrieve member details from the members file and display information on editable fields.  **Step 7:** Validate entered information:   * Surname (Varchar2) * Forename (Varchar2) * Address(   House Number (Varchar2)  Street (Varchar2)  Town (Varchar2)  County (Varchar2)  Eircode (Varchar2)  )   * Phone(numeric) * Email(Varchar2) * Phone(numeric) * Email(Varchar2).   **Step 8:** Save updated information in the Members file.  **Step 9:** Display Confirmation.  **Step 10:** Reset UI. |
| **Alternate Scenario** | **Member** | **System** |
| **Invalid MemberID** |  | **Step 5:** Validation failed.  **Step 6:** Display appropriate error message.  **Step 7:** Return to Step 4. |
| **Invalid Data entered at Step 9** |  | **Step 11:** Validation failed.  **Step 12:** Display appropriate error message.  **Step 13:** Return to Step 10. |
| **Conclusions** | A Member has Updated information into the Members file. | |
| **Post conditions** | The Member’s information can be updated as long as they have a valid MemberID. | |
| **Business Rules** | Must have a Valid MemberID. | |

### Pay Fee

This function allows a Member to pay an existing Fee. A fee is generated when a book is returned past its due date. The Fee increments by a standard fee of €5 per late return. This will be an interaction between the Member and the System.



Member

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Pay Fee | |
| **Use Case Id** | 5 | |
| **Priority** | 1 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Member | |
| **Other Participating Actors** |  | |
| **Description** | This function will be used to pay a members late fee. | |
| **Preconditions** | The Member must have a late fee applied to their account. | |
| **Trigger** | A Member would like to pay a late fee. | |
| **Expected Scenario** | Member | **System** |
|  | **Step 1:** Start Pay Fee function.  **Step 3:** Enter Member Information.   * MemberID                         **Step 11:** Input Bank details.  **Step 16:** Confirm OK. | **Step 2:** Display UI.    **Step 4:** Validate entered data:  Check that the entered data has an entry in the Members file in the database by matching MemberID.    **Step 5:** Retrieve all data from the Members file where MemberID = MemberID entered in step 3.  **Step 7:** Check if FeeAmount > 0.  **Step 8:**  Display UI with Fee Amount.  **Step 10:** Display UI to enter Bank details:   * Bank Card(numeric, must be 16 digits) * CVV(numeric, must be 3 digits)   .  **Step 12:** Validate both entry fields:   * Bank card must be 16 digits * CVV must be 3 digits   **Step 13:** Subtract Fee Amount from BankCard.  **Step 14:** Reset FeeAmount to 0.  **Step 15:** Display Confirmation Message.  **Step 17:** Reset UI. |
| **Alternate Scenario** | **Member** | **System** |
| **Invalid MemberID** |  | **Step 4:** Validation failed.  **Step 5:** Display appropriate error message.  **Step 6:** Return to Step 4. |
| **Invalid Bank details** |  | **Step 12:** Validation failed.  **Step 13:** Display appropriate error message.  **Step 14:** Return to Step 11. |
| **Fee amount = 0** |  | **Step 7:** Fee = 0;  **Step 8:** Display Appropriate Message.  **Step 9:** Return to Step 1. |
| **Conclusions** | A Member has paid their fee. | |
| **Post conditions** |  | |
| **Business Rules** | Must have a Valid MemberID and BankCard. | |

## Books

This module will catalogue new books and allow members to search for books, it also allows members to add books to a cart and then checkout through the search function.

### Catalogue Book

This function will be used to add a new book to the Books file in the database.

This will be an interaction between the Librarian and the System.



Librarian

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Catalogue Books | |
| **Use Case Id** | 6 | |
| **Priority** | 1 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** |  | |
| **Description** | This function will be used to add a new Book to the Books file in the database. | |
| **Preconditions** |  | |
| **Trigger** | A new book needs to be catalogued. | |
| **Expected Scenario** | **Customer** | **System** |
|  | **Step 1:** Start Catalogue Book function.  **Step 3:** Enter Book Information.   * Title * Author * Release Date * Genre * ISBN                             **Step 8:** Confirm OK. | **Step 2:** Display UI.    **Step 4:** Validate data submitted, All fields must have entries:   * Title(varchar2) * Author(varchar2) * Release Date(DATE dd/mmm/yyyy) * Genre(varchar2)   **Step 5:** Generate next available BookID.  **Step 6:** Adds new Book information the Books file in the database:   * Title(varchar2) * Author(varchar2) * Release Date(DATE dd/mmm/yyyy) * Genre(varchar2) * Status(Available(A)/Unavailable(U),   Default = A)  **Step 7:** Display Confirmation message.  **Step 9:** Reset UI. |
| **Alternate Scenario** | **Customer** | **System** |
| **Invalid data** |  | **Step 4:** Validation failed.  **Step 5:** Display appropriate error message.  **Step 6:** Return to Step 3. |
| **Conclusions** | A new Book has been added to the Books file in the database. | |
| **Post conditions** | The Book can now be searched for and can be loaned to Valid Members. | |
| **Business Rules** |  | |

### Search Catalogue

This function is used to check if a book is in the Books file, it allows the user to select which of the following terms to search by: Title, Author, Genre. It also allows the user to add books to a cart, which can then be checked out by calling the record loan function in the background. Items can also be removed from the cart.

This is an interaction between the Member or Librarian and the System.



Librarian

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Search Catalogue | |
| **Use Case Id** | 7 | |
| **Priority** | 1 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Member | |
| **Other Participating Actors** | Librarian | |
| **Description** | This function searches for a book, adds and removes books from a cart, and refers to record loan to check out books. | |
| **Preconditions** | Must have a valid MemberID or be the Librarian. | |
| **Trigger** | A Member or Librarian searches for a book. | |
| **Expected Scenario** | **Customer** | **System** |
|  | **Step 1:** Start search catalogue function.  **Step 3:** Select search filter from dropdown box.  **Step 5:** Enter text to search for.  **Step 9:** Click on the required book.  **Step 12:** Click on checkout button. | **Step 2:** Display UI with dropdown box to select search term.    **Step 4:** Display input field.  **Step 6:** Search Books file where:  Search term = Entered text.  **Step 7:** Search Books file where:  Search term = Entered text AND  Book Status = ‘A’.  **Step 8:** Display all Books information returned from step 7 in a table;   * Book id * Book Title * Author * Release date * Genre * Status   **Step 10:** Display a cart table and add book clicked on to the table;   * Book id * Book Title * Author * Release date * Genre * Status   **Step 11:** Display checkout Button.  **Step 13:** Start Record Loan function.  **Step 14:** Display Confirmation message. |
| **Alternate Scenario** | **Customer** | **System** |
| **Book is not found** |  | **Step 4:** Search Books file where:  BookName = Entered Book name.  **Step 5:** If there is no matching entry found in Step 4, Display empty table. |
| **All items removed from cart** | **Step 12:** Click on last book in cart to remove. | **Step 13:** Hides checkout button. |
| **Conclusions** | A Member can search for books. | |
| **Post conditions** | The Book can be searched for and added to a cart by members. | |
| **Business Rules** |  | |

|  |  |
| --- | --- |
| User | System |
| Click on Book to add to cart  Click Search button  Type search parameters  Select search Category  Start Search Catalogue function | Display checkout button  Display cart box and display clicked-on book details  Display retrieved Books on UI  Retrieve all data from the books file where category and parameters match, and status is available  Display search bar with appropriate label  Load and Display UI |

|  |  |
| --- | --- |
| User | System |
| Enter member id  Click on checkout button | Create a new loan instance  Check if member has < 3 strikes  [ N ]  Display Error Message  < 3?  [ Y ]  Check if member has < 3 strikes  [ N ]  Valid?  Check is member id valid  Display Member id input box |

## Loans

This module will allow loans to be recorded and books to be returned. It will update the status of books returned to be available and close a loan once all items have been returned.

### Record Loan

This function will be used to add a new loan instance to the Loans file, it is called during the search catalogue function when a member clicks on checkout.

This will allow Members to be the recipient of loaned Books. It is a background function.



Librarian

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Record Loan | |
| **Use Case Id** | 8 | |
| **Priority** | 1 | |
| **Source** | Search Catalogue | |
| **Primary Business Actor** | Member | |
| **Other Participating Actors** |  | |
| **Description** | This function will be used to Loan a Book to a Member. | |
| **Preconditions** | Book must be available. | |
| **Trigger** | Checkout button clicked in search catalogue. | |
| **Expected Scenario** | Member | **System** |
|  | **Step 1:** Click on Checkout button.  **Step 3:** Enter Member information:   * MemberID | **Step 2:** Display Input box for Member ID.    **Step 4:** Validate data submitted, check that entered MemberID has a matching value in the Members file.    **Step 5:** Generate next Loan id.  **Step 7:** Create new Loan:   * Loan ID(numeric, 5) * Member ID(numeric, 5) * DueDate(VarChar2, 10) * Status(VarChar2, 1, Default = ‘O’)   **Step 8:** For each item in the ‘Search Catalogue’ cart, create a LoanItem:   * Loan ID(numeric, 5, same id generated in step 5) * Book ID(Numeric, 5, book id from current cart item) * GenreCode(Varchar2, 3, GenreCode from current cart item) * ReturnDate(VarChar2, 10, null) * Status(VarChar2, 1, Default = ‘L’)   **Step 9:** Change Book Status to “U”  In the Books file for the current cart item.  **Step 10:** Display Confirmation message.  **Step 12:** Back to Search Catalogue. |
| **Alternate Scenario** | Librarian | **System** |
| **Invalid data** entered at step 3 |  | **Step 4:** Validation failed.  **Step 5:** Display appropriate error message.  **Step 6:** Return to Step 3. |
| **Conclusions** | A new Loan instance has been added to the Loans file, new LoanItems have been added to the LoanItems file and the member has books checked out. | |
| **Post conditions** | The Loan will change the status of the book and will apply the loan period to the reservation. | |
| **Business Rules** |  | |

### View Loan

This function will allow Members to view Loans, to check due dates for loaned items. It takes a valid member id and returns any loaned items.

.



Member

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | View Loan | |
| **Use Case Id** | 9 | |
| **Priority** | 1 | |
| **Source** | Member | |
| **Primary Business Actor** | Member | |
| **Other Participating Actors** |  | |
| **Description** | This function will be used to view a Loan. | |
| **Preconditions** | Member must have . | |
| **Trigger** | Member wants to Return Book. | |
| **Expected Scenario** | **Librarian** | **System** |
|  | **Step 1:** Start Return Book function.  **Step 3:** Enter Member information:   * MemberID | **Step 2:** Display UI.    **Step 4:** Validate data submitted, check that entered MemberID has a matching value in the Members file.  **Step 5:** Retrieve Loan id data from the Loans file where MemberID entered in step 3 = MemberID in loans file.  **Step 6:** Get all LoanItems where Loan ID = Loan ID retrieved in Step 5.    **Step 7:** For each LoanItem, Get all book information where book id = book id in LoanItem.  **Step 8:** Display all book information on a Loans Table. |
| **Conclusions** | | A Book has been returned and has been made available to be loaned to other members. Late fees have been applied where necessary to a Member’s account | |
| **Post conditions** | | Other Members can Borrow the book. | |

### Return Book

This function will allow Members to return Books, updating book status.

This also updates the LoanItem status to ‘R(Returned)’ and if all associated LoanItems have a status of ‘R’, the Loans status updates to ‘C(Closed)’.

This will allow Members to return Books to avoid paying a late fine and will make the Book available for other Members.



Member

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Return Book | |
| **Use Case Id** | 10 | |
| **Priority** | 1 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** | Member | |
| **Description** | This function will be used to return a Book to the catalogue. | |
| **Preconditions** | Book must be loaned to Member. | |
| **Trigger** | Member wants to Return Book. | |
| **Expected Scenario** | **Librarian** | **System** |
|  | **Step 1:** Start Return Book function.  **Step 3:** Enter Member information:   * MemberID             **Step 9:** Click on a book to return.  **Step 11:** Confirm. | **Step 2:** Display UI.    **Step 4:** Validate data submitted, check that entered MemberID has a matching value in the Members file.  **Step 5:** Retrieve Loan id data from the Loans file where MemberID entered in step 3 = MemberID in loans file.  **Step 6:** Get all LoanItems where Loan ID = Loan ID retrieved in Step 5.    **Step 7:** For each LoanItem, Get all book information where book id = book id in LoanItem.  **Step 8:** Display all book information on a Loans Table.  **Step 10:** Show Confirmation Message.  **Step 12:** Get LoanItem where MemberID = MemberID in Step 3 AND BookID = BookID in book clicked.  **Step 13:** Update the LoanItem ReturnDate, Set = Current date.  **Step 14:** Update Book Status, Set it to ‘A’  **Step 15:** Update LoanItem Status, Set it to ‘R’  **Step 16:** Check LoanItem ReturnDate < Loan DueDate.  **Step 17:** Check if each Loan’s LoanItems status is ‘R’.  **Step 18:** Update Loan Status = ‘C’.  **Step 19:** Check if Member StrikeCount >= 3.  **Step 20:** Show Confirmation message. |
| **Alternate Scenario** | Librarian | **System** |
| ReturnDate is > DueDate at Step 16 |  | **Step 17:** Add fee(€5.00) to Fee Amount in Members file where MemberID retrieved in Step 5 = MemberID in Members file.  **Step 18:** Increase Strike Count by 1 in Members file where MemberID retrieved in Step 5 = MemberID in Members file.  **Step 19:** Display Pay Fee button. |
| Step 19 returns StrikeCount >= 3 is true |  | **Step 20:** Display appropriate Message. |
| **Conclusions** | A Book has been returned and has been made available to be loaned to other members. Late fees have been applied where necessary to a Member’s account | |
| **Post conditions** | Other Members can Borrow the book. | |
| **Business Rules** |  | |

## Admin

This Module gives access to real time analysis of which genre is most popular, allowing for better use of library resources. It will be presented through the use of a GUI Chart.

### Review Loans

This function allows analysis of which genre is most popular, allowing for better use of library resources. It will be presented through the use of a GUI Chart.



Librarian

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Review Loans | |
| **Use Case Id** | 11 | |
| **Priority** | 3 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** |  | |
| **Description** | This function will be used to review Book Loans and get Genre Popularity. | |
| **Preconditions** |  | |
| **Trigger** | Librarian wants to review loaned Books popularity. | |
| **Expected Scenario** | **Librarian** | **System** |
|  | **Step 1:** Start Review Loan function. | **Step 2:** Retrieve all LoanItems.  **Step 3:** For each LoanItem, get GenreCode.  **Step 4:** Utilise a Counter for each GenreCode that adds 1 to the individual counter for each LoanItem with a matching GenreCode.  **Step 5:** Display GUI Chart for analysis, a pie chart with sections and genre identifiers. |
| **Conclusions** | The Librarian can view all books previously loaned. | |
| **Post conditions** |  | |
| **Business Rules** |  | |

A pie chart with text on it

Description automatically generated

*Example of Genre Popularity chart*

# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

LibrarySYS

Member

Return Book

Loan Book

## Level-1 DFD

P1

Manage

Members

Details

Details

Member

Members File

D1

Membership status

Member

P3

Control

Loans

P4

Perform

Admin

P2

Manage

Books

Librarian

Loans File

D3

Books File

D2

Request Loan Details

Return Loan Details

Request book/s

New Book Details

Genres File

D2

Book and genre Details

Genre details

Loan Item Details

## Level-2 DFD(Process P1: Members)

P1

Register

Members

P2

View

Member

Details

Member

Details

Membership

Status

Members File

D1

Member Details

Request Member Details

Request Member Details

P3

Update

Member

Information

P4

Pay Fees

Members File

D1

Revoked Membership

New Member

Details

New Member

Details

Fee Query

Fee amount

Fee Amount Paid

P3

Remove Members

Member Details

Updated status

Librarian

Member

Pay fee

Librarian

## Level-2 DFD(Process P2: Books)

P1

Catalogue Books

P2

Search Catalogue

Member

Books File

D2

Book Details

Librarian

New Book Details

Book details

Search Book details

Return Book details

Return Book details

L1

Record Loan

Add to loans

## Level-2 DFD(Process P3: Loans)

P1

Record Loan

P2

Return Book

Member

Loans File

D3

Book Details

Requested Book Details

Book details

Returned Book details

B2

Search Catalogue

## Level-2 DFD(Process P4: Admin)

P1

Review Loans

Loans File

D3

All Loan Details

Librarian

Request Loans Review

Sorted Loan Details

Request All Loan Details

LoanItems File

D2

Get Genres count

Return Genres Count

# Data Model

The next section is the data model section. This section will include the Class Diagram, the relational schema and the Database Schema. The Class Diagram will be in UML notation and the Relational Schema will be in Bracket Notation. It will briefly describe the relation between each table.

## Class Diagram

Member

-MemberID\*

-Forename

-Surname

-Address

-Status

-Phone

-Email

-Status

-FeeAmount

-StrikeCount

Loan

-LoanID\*

MemberID

-DueDate

-Status

Takes out a

1..1

\*..1

Consists of

1..\*

\*..1

LoanItem

-LoanID

-BookID

-GenreCode

-ReturnDate

-Status

Is tied to

1..1

1..1

Genres

-GenreCode

1..1

1..1

Book

-BookID\*

-Title

-Author

-ReleaseDate

-Status

-GenreCode

-Description

Has a

1..\*

1..\*

Is tied to

## Relational Schema

Member(MemberID, Forename, Surname, Address, Status, Phone, Email, Status, FeeAmount, StrikeCount)

Book(BookID, Title, Author, ReleaseDate, GenreCode, Status)

Loan(LoanID, MemberID, DueDate, Status)

LoanItem(LoanID, BookID, GenreCode, ReturnDate, Status)

Genre(GenreCode, Description)

## Database Schema

This schema is based on a library system that has 5 tables.

**Relation Members**

MemberID numeric(5) NOT NULL,

ForeName varchar2(20) NOT NULL,

SurName varchar2(20) NOT NULL,

HouseNo varchar2(3) NOT NULL,

Street varchar2(20) NOT NULL,

Town varchar2(20),

County varchar2(20) NOT NULL,

Eircode varchar2(7) NOT NULL,

PhoneNo varchar2(10) NOT NULL,

Email varchar2(50),

Status varchar2(1) DEFAULT 'A',

FeeAmount numeric(3) DEFAULT 0,

StrikeCount numeric(1) DEFAULT 0,

**PRIMARY KEY:** MemberID

**Relation Genres**

GenreCode VARCHAR2(3) NOT NULL,

Descript VARCHAR2(15) NOT NULL,

**PRIMARY KEY:** GenreCode

**Relation Books**

BookID numeric(5) NOT NULL,

Title varchar2(35) NOT NULL,

Author VARCHAR2(40) NOT NULL,

ReleaseDate VARCHAR2(11) NOT NULL,

GenreCode VARCHAR2(3) NOT NULL,

Status VARCHAR2(1) DEFAULT 'A',

**PRIMARY KEY:** BookID

**FOREIGN KEY:** GenreCode **REFERENCES** Genres

**Relation Loans**

LoanID numeric(5) NOT NULL,

MemberID numeric(5) NOT NULL,

DueDate VARCHAR2(10) NOT NULL,

Status VARCHAR2(1) DEFAULT 'O',

**PRIMARY KEY:** LoanID

**FOREIGN KEY:** MemberID **REFERENCES** Members

**Relations LoanItems**

LoanID numeric(5) NOT NULL,

BookID numeric(5) NOT NULL,

GenreCode VARCHAR2(3) NOT NULL,

ReturnDate VARCHAR2(10),

Status VARCHAR2(1) DEFAULT 'L',

**PRIMARY KEY:** LoanID, BookID

**FOREIGN KEY:** LoanID **REFERENCES** Loans

**FOREIGN KEY:** BookID **REFERENCES** Books

**FOREIGN KEY:** GenreCode **REFERENCES** Genres

# Conclusion

In conclusion, this library system is a basic system that has a balance of user functionality, system upkeep and admin reviewal. The main objective of this system is the management of members and books, with focus on loaning books to members and adding new books to the catalogue. There is also a Strike count which keeps track of late returns, incurring a fee for any item returned late. Finally there is administrative application through the use of a function that reviews genre popularity to allow management to buy more books of a popular genre.