Requirements Engineering

LibrarySYS

Submitted By: Daragh Carroll (t00201097)

Computing with Games Development

Date Submitted: dd/mm/yyyy

**Table of Contents**

[1. Introduction/overview 4](#_Toc528311901)

[2. Functional Components 5](#_Toc528311902)

[3. User Requirements 6](#_Toc528311903)

[4. System Requirements 7](#_Toc528311904)

[4.1. System Level Use Case Diagram 7](#_Toc528311905)

[4.2. Manage Members 8](#_Toc528311906)

[4.2.1. Add member 9](#_Toc528311907)

[4.2.2. Update member 11](#_Toc528311908)

[4.2.3. De-register member 13](#_Toc528311909)

[4.3. Books 15](#_Toc528311910)

[4.3.1. Add book 15](#_Toc528311911)

[4.3.2. Update book 17](#_Toc528311912)

[4.3.3. Remove book 19](#_Toc528311913)

[4.3.4. Check book availability 20](#_Toc528311921)

[4.4. Rentals 22](#_Toc528311922)

[4.4.1. Borrow book 22](#_Toc528311924)

[4.4.2. Return book 26](#_Toc528311925)

[4.4.3. List overdue books 28](#_Toc528311926)

[4.4.4. Pay Fine 30](#_Toc528311935)

[4.5. Admin 32](#_Toc528311936)

[4.5.1. Analyse Loans by Genre 32](#_Toc528311943)

[4.5.2. Analyse Members 34](#_Toc528311944)

[5. System Model 36](#_Toc528311945)

[5.1. Level-0 DFD 36](#_Toc528311946)

[5.2. Level-1 DFD 37](#_Toc528311947)

[5.3. Level-2 DFD (Process P1: Title) 38](#_Toc528311948)

[5.4. Level-2 DFD (Process P2: Title) 39](#_Toc528311949)

[5.5. Level-2 DFD (Process P3: Title) 40](#_Toc528311950)

[6. Data Model (Class Diagram) 42](#_Toc528311951)

[6.1. Class Diagram 42](#_Toc528311952)

[6.2. Relational Schema 42](#_Toc528311953)

[6.3. Database Schema 42](#_Toc528311954)

[7. Conclusion 43](#_Toc528311955)

[8. Appendices 44](#_Toc528311956)

[8.1. Appendix A – Title 44](#_Toc528311957)

[8.2. Appendix B – Title 44](#_Toc528311958)

# Introduction/overview

**Normal** text here

# Functional Components

# User Requirements

High level **abstract statements** describing the user requirements.

This should be consistent with the hierarchy chart in section 2.

Requirements to be listed in exactly same order as in section 2.

1. LibrarySYS will manage library membership.
   1. LibrarySYS will add a new member.
   2. LibrarySYS will update a members details.
   3. LibrarySYS will de-register a patient
2. LibrarySYS will perform book administration.
   1. LibrarySYS will register a new book.
   2. LibrarySYS will update books’ details.
   3. LibrarySYS will remove a book.
   4. LibrarySYS will check the availability of a particular book.
3. LibrarySYS will manage book rentals
   1. LibrarySYS will rent books out to a member.
   2. LibrarySYS will return a book from a member.
   3. LibrarySYS will list overdue books.
   4. LibrarySYS will record the payment of fines.
4. LibrarySYS will perform administrative reporting.
   1. LibrarySYS will analyse loans by genre.
   2. LibrarySYS will analyse member demographics.

# System Requirements

This section of the document presents the abstract user requirements as low-level system requirements.

## System Level Use Case Diagram

The high-level software modules are represented on the system level use case diagram shown below.

Librarian

Member

## Manage Members

The module provides functions to allow the management of library members. The functions provided include registering, updating and de-reregistering a member. Each member is identified by a unique member id which is assigned by the system when a member is created.

### Add member

Each member of the library is identified by a unique member Id which is assigned by the system. This function captures, validates and stores the member details.

Member

Librarian

<<Extends>>>

<<Extends>>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add Member | |
| **Use Case Id** | 1 | |
| **Priority** | 1 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** | Member | |
| **Description** | This function adds a new member to the library system | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The librarian invokes the add member function  **Step 3:** The librarian enters the member’s details:   * Surname * Forename * Date of Birth * Street * Town * Phone number * email   **Step 4:** The librarian confirms that the member is to be added. | **Step 2:** The System displays the UI  **Step 5:** The System validates the  entered information:   * All fields must be entered * Name, Forename, Town must not be numeric * Date of Birth must not be a future date * The email address must be a valid format   **Step 6:** The system determines the next MemberID  **Step 7:** The System adds a value of 0 for the member’s late fees.  **Step 8:** The System saves the member’s status to ‘a’ for “Active”.  **Step 9:** The System saves the new book in the ***Members file***.   * MemberID * Surname * Forename * Date of Birth * Gender * Phone number * Email * Late fees = 0 * Status = “Active”   **Step 10:** The System displays a confirmation message  **Step 11:** The System clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 5:** Invalid data is entered.  **Step 6:** An appropriate error message is displayed. |
| **Conclusions** | A new member is added to the Members file. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update member

A member’s details can be updated by the Librarian at the request of the member in question.  
The member must have an active library card to change their details. This function checks the member’s activity status and captures the new details to make the changes.

Librarian

Member

<<Extends>>>

<<Extends>>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update member | |
| **Use Case Id** | 2 | |
| **Priority** | 2 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** | Member | |
| **Description** | This function changes the information attached to a member account | |
| **Preconditions** | Member must already exist, Member must be active | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The librarian invokes the update member function  **Step 3:** The librarian enters the Surname (or part of) of the member they wish to alter.  **Step 5:** The librarian selects the member they wish to alter from the list provided by the System.  **Step 7:** The librarian makes the required alterations:   * Surname * Forename * Date of Birth * Street * Town * Phone number * Email   **Step 8:** The librarian confirms that the member is to be updated. | **Step 2:** The System displays the UI.  **Step 4:** The System retrieves a summary of all active members with matching Surname fromthe ***Members file*** and displays on UI.  **Step 6:** The System retrieves all details for the selected member from the ***Members file*** and displays on UI for editing.    **Step 9:** The System validates the member   * All fields must be entered * Name, Forename, Street, Town must not be numeric * Date of Birth must not be a future date * The email address must be a valid format   **Step 10:** The System updates the new information in the ***Members file***   * Surname * Forename * Date of Birth * Street * Town * Phone number   **Step 11:** The System displays a confirmation message  **Step 12:** The System clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | Any altered information overwrites the old information in the **Members file**. | |
| **Post conditions** |  | |
| **Business Rules** | Only active members can be updated | |
| **Implementation Constraints** |  | |
|  |  | |

### De-register member

The Librarian can make a member inactive if they do not renew their library card. This function allows the librarian to make a member inactive in the members file.

Librarian

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | De-register member | |
| **Use Case Id** | 3 | |
| **Priority** | 4 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** |  | |
| **Description** | This function makes a member account inactive. | |
| **Preconditions** | Member must already exist. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The librarian invokes the de-register member function  **Step 3:** The librarian enters the Surname (or part of) of the member they wish to de-register.  **Step 5:** The librarian selects the member they wish to de-register from the list provided by the System.  **Step 6:** The librarian confirms that the member is to be made inactive. | **Step 2:** The System displays the UI.  **Step 4:** The System retrieves a summary of all active members with matching Surname fromthe ***Members file*** and displays on UI.    **Step 7:** The System updates the ***Members file*** so that it is In-active***.***  **Step 7:** The System displays a confirmation message  **Step 8:** The System clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The member becomes inactive in the Members file. | |
| **Post conditions** |  | |
| **Business Rules** | Only active members can be made inactive | |
| **Implementation Constraints** |  | |

## Manage Books

Each book in the library is identified by a unique book Id which is assigned by the system. This function captures, validates and stores the book details.

### Add book

The Librarian can add a new book containing a title, genre, author and ISBN.

Librarian

<<Extends>>>

<<Extends>>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add Book | |
| **Use Case Id** | 4 | |
| **Priority** | 1 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** |  | |
| **Description** | This function adds a new book to the library. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The librarian invokes the add book function  **Step 3:** The librarian enters the book’s details:   * Title * Genre * Author * ISBN   **Step 4:** The librarian confirms that the book is to be added. | **Step 2:** The System retrieves details of book genres from the ***Genres File*** and displays UI  **Step 5:** The System validates the entered information:   * All text fields must be filled * ISBN must be 13 to 15 numbers long   **Step 6:** The System allocates the book the next available BookID.  **Step 8:** The book’s availability status is set to available in the ***Books file***.  **Step 7:** The System saves the new book in the ***Books file***.   * BookID * Title * Genre * Author * ISBN * Status   **Step 9:** The System displays a confirmation message  **Step 10:** The System clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 6:** Invalid data is entered.  **Step 7:** An appropriate error message is displayed. |
| **Conclusions** | A new book is added to the *Books file*. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update book

This function captures new information about a book and overwrites the old information.

Librarian

<<Extends>>>

<<Extends>>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Book | |
| **Use Case Id** | 5 | |
| **Priority** | 2 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** |  | |
| **Description** | This function changes information in the Book File | |
| **Preconditions** | The book must already exist | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The librarian invokes the update book function  **Step 3:** The librarian enters the title (or part of the title) of the book they would like.  **Step 5:** The librarian selects which book they would like to alter from the list of books.  **Step 7:** The librarian enters the book’s details:   * Title * Genre * Author * ISBN   **Step 8:** The librarian confirms that the book is to be altered. | **Step 2:** The System displays the UI.  **Step 4:** The System displays a summary of books with titles like that of the provided search terms from the ***Books file***.  **Step 6:** The System displays the information of the selected book from the ***Books file.***  **Step 9:** The System validates the entered information:   * All text fields must be filled * ISBN must be between 12-15 numbers   **Step 10:** The System updates the new book information in the ***Books file***.  **Step 11:** The System displays a confirmation message  **Step 12:** The System clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 9:** Invalid data is entered.  **Step 10:** An appropriate error message is displayed. |
| **Conclusions** | Pre-existing information about the book is changed in the Books file. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Remove book

This function allows the Librarian to remove an available book from the Books file

Librarian

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Remove Book | |
| **Use Case Id** | 6 | |
| **Priority** | 4 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** |  | |
| **Description** | This function removes a book from Books file. | |
| **Preconditions** | Book must already exist. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The librarian invokes the remove book function  **Step 3:** The librarian enters the title of the book they would like to remove  **Step 5:** The librarian selects which book they would like to remove from the list  **Step 7:** The librarian confirms that the book is to be removed. | **Step 2:** The System displays the UI  **Step 4:** The System retrieves a list of the books with titles the same as or similar to the title searched.  **Step 6:** The System requests confirmation that the book is to be removed  **Step 8:** The System removes the selected Book from the ***Books file***  **Step 9:** The System clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The book is removed from the Books file. | |
| **Post conditions** |  | |
| **Business Rules** | Only available books can be removed | |
| **Implementation Constraints** |  | |



### Check book availability

This function allows the Librarian to check if a book is available for rental.

Librarian

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Check book availability | |
| **Use Case Id** | 7 | |
| **Priority** | 5 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** |  | |
| **Description** | This function checks if a book is available. | |
| **Preconditions** | Book must already exist. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The librarian invokes the check book availability function  **Step 3:** The librarian inputs the title of the book they are checking.  **Step 5:** The librarian confirms that they are finished checking available books | **Step 2:** The System displays the UI  **Step 4:** The System displays a list of available books from the ***Books file*** with a title the same as or similar to the one entered.  **Step 6:** The System clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  | **Step 4:** The System searches for the book in the ***Books file****.*  **Step 5:** The System displays an appropriate message indicating that the book searched was not found. |
| **Conclusions** | The System indicates whether a book is available at present. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Manage Rentals

This module includes functions for renting out a book, returning a book, generating a list of overdue books and for the payment of fines. Each rental is identified by a unique Rental ID which is assigned by the system.



### Borrow book

This function captures what member is renting books, which books they are renting, the date they are renting it and the day it is to be returned by and adds it to the Rentals file. Each rental has an ID which is provided by the system.

Librarian

Member

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Borrow Book | |
| **Use Case Id** | 8 | |
| **Priority** | 3 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** | Member | |
| **Description** | This function makes selected books unavailable and adds it to a member’s account | |
| **Preconditions** | Member must be active. | |
| **Trigger** | The User takes a book(s) to be borrowed. | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:**The librarian invokes the borrow book function  **Step 3:** The librarian inputs the title of the book they would like to rent out  **Step 5:** The librarian selects what book is being rented from the provided list.  **Step 7:** The librarian confirms that the book is to be added to the rental  **Step 9:** The librarian enters the surname of the member they would like to add to the rental.  **Step 11:** The librarian selects which member they would like from the list.  **Step 14:** The librarian confirms that the rental is to be added. | **Step 2:** The System displays the new rental UI  **Step 4:** The System displays a list of available books with a title the same as or similar to the one entered from the ***Books file.***  **Step 6:** The System requests confirmation from the user  **Step 8:** The book is added to a ‘cart’ of books.  **Step 10:** The System displays a list of members with the same or a similar surname to that entered from the ***Members file***  **Step 12:** The System checks to see if the member has any outstanding late fees.  **Step 13:** The System displays an appropriate confirmation message  **Step 15:** The System adds the member to the rental in the ***Rentals file***  **Step 16:** The System assigns the next RentalID to a new rental in the ***Rentals file***  **Step 17:** The System adds the Due Date to the ***Rentals file***.   * System\_Date + 14 days   **Step 18:** The book ID is added from the ***Books file*** to the ***RentalItems file***  **Step 19:** The rental date is added to the ***RentalItems******file*** as the current system date.  **Step 20:** The Rental item from the ***RentalItems file*** is contained by the new rental in the ***Rentals file***.  **Step 21:** The book is made unavailable in the ***Books file***  **Step 22:** The System clears the UI |
|  | **Actor Action** | **System Response** |
|  | **Step 3:** The librarian inputs the title of the book they would like to rent out  **Step 9:** The librarian enters the surname of the member they would like to add to the rental.  **Step 11:** The librarian selects which member they would like from the list.  **Step 14**: The librarian does not confirm the rental in the confirm rental dialog. | **Step 4:** The System displays an appropriate error message to indicate that no book was found.  **Step 10:** The System displays an appropriate error message to indicate that no member was found  **Step 12:** The System checks to see if the member has any outstanding late fees.  **Step 13:** The System displays an appropriate error message dictating that the selected member cannot rent any books until they pay their late fees  **Step 15:** The System clears the UI. |
| **Conclusions** | The books file is altered to indicate that the book is unavailable. The book and member information is saved to the rentals file. | |
| **Post conditions** |  | |
| **Business Rules** | A member cannot borrow a book if they have outstanding late fees | |
| **Implementation Constraints** |  | |

Create rental

**Librarian**

**System**

Borrow Book

Error Message

Input title

Display UI

Titles  
Found

**N**

**Y**

Select book(s)

Show books

Add book to  
books cart

Input Member Surname

Error Message

Member  
Found

**N**

**Y**

Select member

Show books

### Return book

This function makes the book available again in the Book file and sets the outstanding rental in the Rentals file to complete, adding late fees if required.

Librarian

Member

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Return Book | |
| **Use Case Id** | 9 | |
| **Priority** | 3 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** | Member | |
| **Description** | This function makes the book available and sets the Rental status to be complete. It also calculates late fees to be added to the member if necessary | |
| **Preconditions** |  | |
| **Trigger** | The Member returns a book | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:**The librarian invokes the return book function  **Step 3:** The librarian inputs the surname of the member they wish to see.  **Step 5:** The librarian selects which member they want to view.  **Step 7:** The librarian selects which rental is to be closed.  **Step 9:** The librarian confirms the change. | **Step 2:** The System displays the UI.  **Step 4:** The System displays a list of the members with a surname the same as or similar to that entered from the ***Members file***  **Step 6:** The System displays all of the member’s ongoing rentals.  **Step 8:** The System requests confirmation from the user.  **Step 10:** The System alters the ***Books file***.   * Availability = “A”   **Step 11:** If the Due date in the ***Rentals file*** has passed, the System will calculate the late fees.  **Step 12:** These late fees are added to the member in the ***Members file.***  **Step 13:** The System displays a confirmation message  **Step 14:** The System clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The book file is altered so that the book is marked as available. Any late fees are calculated and added to the member’s account in the member file. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### List overdue books

This function allows the librarian to list any books that are past their due date from the Rentals file.

Librarian

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | List overdue books | |
| **Use Case Id** | 10 | |
| **Priority** | 5 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** |  | |
| **Description** | This function lists all books that are past their due date. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The librarian invokes the list overdue books function.  **Step 4:** The librarian enters the date they wish to view.  **Step 5:** The librarian confirms the input.  **Step 8**: The librarian confirms that they are done viewing the list. | **Step 2:** The System retrieves the rental information from the ***Rentals file.***  **Step 3:** The System displays the UI.  **Step 6:** The System gets the book title and book id from the **Rentals file** where the due date was before the date entered.  **Step 7:** The System displays those books in list format.  **Step 9:** The System clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The system displays any books in the rentals file that are overdue. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |



### Pay Fine

This function allows the Librarian to remove fines from a member account once they have been paid

Member

Librarian

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Pay Fine | |
| **Use Case Id** | 11 | |
| **Priority** | 3 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** | Member | |
| **Description** | This function removes an allotted amount from the late fees section of the Members file. | |
| **Preconditions** |  | |
| **Trigger** | A member pays their allotted fine. | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The librarian invokes the pay fine function.  **Step 3:** The librarian inputs the surname of the Member whose fines they wish to view.  **Step 5:** The librarian selects which member they would like to view from the provided list.  **Step 8:** The librarian enters the amount the fine is to be reduced by.  **Step 10:** The librarian confirms the change. | **Step 2:** The System displays the UI.  **Step 4:** The System displays a list of all of the members whose surname is the same as or similar to that entered in the ***Members file***  **Step 6:** The System retrieves the fine information from the ***Members file.***   * MemberID * Name * Late fees   **Step 7:** The System displays the member’s information and their present late fees.  **Step 9:** The System requests confirmation from the user.  **Step 11:** The System reduces the Member’s late fees in the ***Members file*** by the amount the user entered.  **Step 12:** The System displays a confirmation message.  **Step 13:** The System clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The member late fees are reduced by the amount the Librarian entered. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Perform Administration

This module contains functions for performing administrative reporting. This includes analysing the number of loans per Genre of book and the number of different members representing different gender or age demographics.



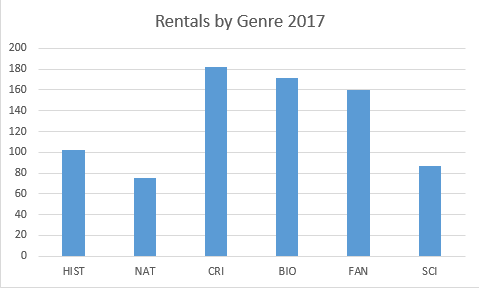
### Analyse Loans by Genre

This function allows the Librarian to analyse the number of the loans members have taken in each genre.

Librarian

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Analyse Loans by Genre | |
| **Use Case Id** | 12 | |
| **Priority** | 6 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** |  | |
| **Description** | This function analyses the number of loans taken out by members in each genre. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The librarian invokes the analyse loans by genre function.  **Step 3:** The librarian selects the year they wish to analyse  **Step 8:** The Librarian confirms that they wish to close the list. | **Step 2:** The System displays the UI  **Step 4:** The System retrieves the rental information from the ***Rentals file.***   * BookID   **Step 5:** The system uses the BookID retrieved from the ***Rentals file*** to retrieve Book information from the ***Books file***.   * GenreID   **Step 6:** The system uses the GenreID in the ***Books file*** to retrieve genre names from the ***Genres file***  **Step 7:** The System counts each time a book of each genre was rented and displays it on a table.  **Step 9:** The System clears the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The system displays the number of books that have been rented in their specific genres. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

**Example graph:**



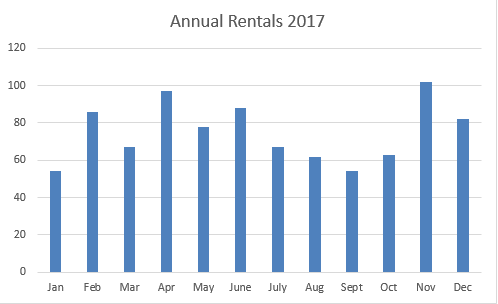
### Analyse Annual Rentals

This function allows the Librarian to analyse the members of the library by age or gender.

Librarian

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Analyse Annual Rentals | |
| **Use Case Id** | 13 | |
| **Priority** | 6 | |
| **Source** | Librarian | |
| **Primary Business Actor** | Librarian | |
| **Other Participating Actors** |  | |
| **Description** | This function analyses the rentals in the different months of a given year | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The librarian invokes the analyse annual rentals function  **Step 3:** The librarian selects what year they would like to analyse  **Step 7:** The Librarian confirms that they wish to close the analysis. | **Step 2:** The System displays the UI  **Step 4:** The System retrieves the rental information from the ***Rentals file*** where the rental year is the same as that entered   * RentalDate   **Step 5:** The System performs a count on the rentals of the different months in the entered year.  **Step 6:** The System displays the count in a graph.  **Step 8:** The System clears the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The system displays the number of rentals in a selected year | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

**Example graph:**



# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

The external entity Member borrows and returns books from and to the Library system

Library  
SYS

Member

Books

Loan Request

## Level-1 DFD

Member Details

Manage  
Members

P1

Member

D1

Members file

Book Details

Book Details

D2

Books file

Rentals Details

D3

Rentals file

Member Details

Loan Request

D4

Rentalitems file

RentalItems  
Details

Member Details

Member Details

Manage  
Rentals

P3

Perform  
Administration

P4

RentalItems Details

Book Details

D4

Rentalitems file

Manage  
Books

P2

Rentals file

Genre Details

Genre Details

D3

Rentals file

D2

Genres file

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

Member Details

Member

Update  
Member

P1.2

Add  
Member

P1.1

Member Details

Member Details

Member Details

Member Details

D1

Members file

Member Details

Activity status

De-register  
Member

P1.3

Withdrawal  
Request

Member

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

Update Book

P2.2

Add Book

P2.1

Book Details

Book Details

Book Details

Book Details

D2

Books file

Book ID

Check Book  
Availability

P2.4

Remove Book

P2.3

Book Details

## Level-2 DFD (Process P3: Manage Rentals)

D2

Books file

D1

Members file

D3

Rentals file

Members Details

Late fees

Availability  
Status

Return Book

P3.2

Book Details

Member Details

Borrow Book

P3.1

Book Details

Member

Rental ID

Member Details

Rental  
 Details

Rental Details

RentalItems  
Details

D3

RentalItems file

RentalItems  
Details

Fine

Member

List Overdue  
Books

P3.3

Rental  
Details

Pay Fine

P3.4

Member  
Details

Late Fees

Member  
Details

D1

Members file

## Level-2 DFD (Process P3: Perform Administration)

Genres  
Details

Rental Items  
Details

Rental Items  
Details

D4

RentalItems file

Rental  
Details

Member  
Details

Analyse Loans by Genre

P4.1

Book  
Details

D3

Rentals file

D5

Genres file

Rental  
Details

Book  
Details

Analyse Annual Rentals

P4.2

D1

Members file

D2

Books file

# Data Model (Class Diagram)

Brief introduction……

## Class Diagram

Object Model – UML Class Diagram

Class diagram shows objects & attributes

Author

ISBN

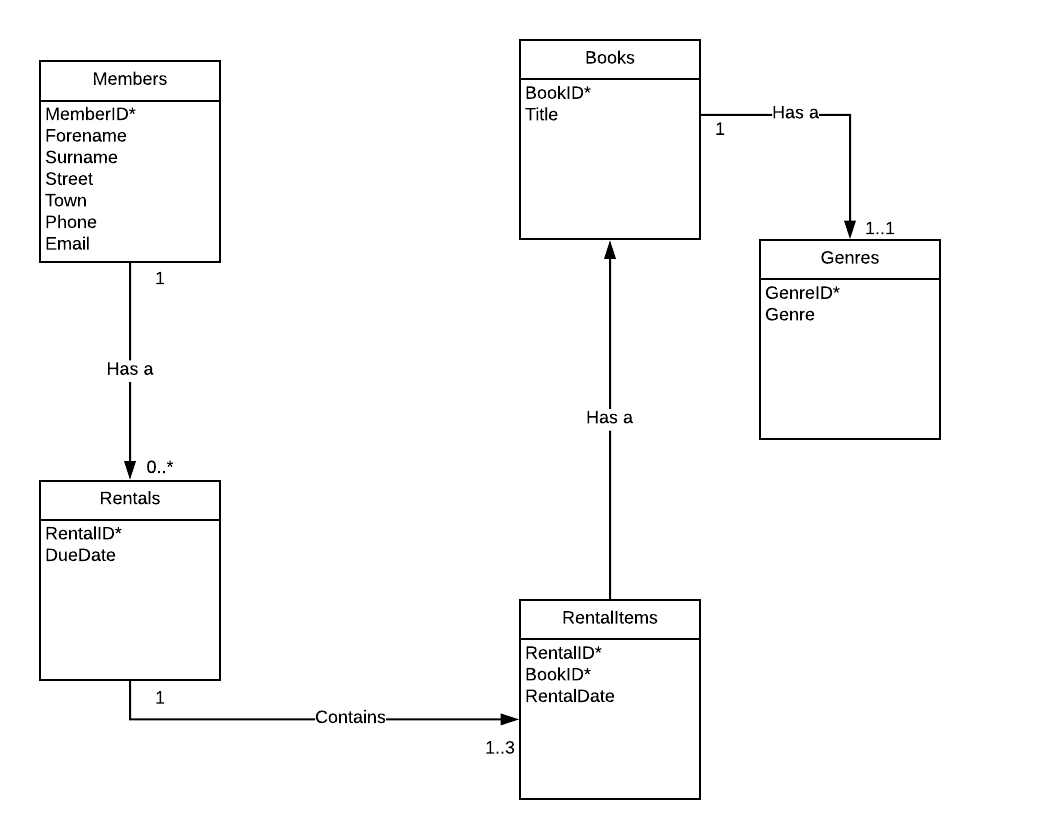
Availability

RentalDate

ReturnDate

Status

MemberId

******

## Relational Schema

Relational schema for the data requirements - Using ***bracket notation***

**Genres(**GenreID, Genre**)**

**Books(**BookID, Title, GenreID, Availability**)**

**Members(**MemberID, Surname, Forename, Street, Town, Phone, Email, Late fees**)**

**Rentals(**RentalID, DueDate, MemberID**)**

**RentalItems(**RentalID, BookID, RentalDate**)**

## Database Schema

A definition of the database to be implemented.

This includes primary key, foreign key and other constraints to be implemented.

**Relation Genres**

GenreID numeric(3)  
 Genre String(15)

**Primary Key:** GenreID

**Relation Books**

BookID numeric(5)  
 GenreID numeric(3)  
 Title String(30)

Availability varchar(1)

**Primary Key:** BookId

**Foreign Key:** GenreID references Genres

**Relation Members**

MemberID numeric(5)

Forename String(15)  
 Surname String(20)

Date of Birth date

Phone String(12)

Email String(25)

Street String(20)

Town String(15)

Late\_Fees numeric(4)

**Primary Key:** MemberID

**Relation Rentals**

RentalID numeric(5)

MemberID numeric(5)

DueDate date

**Primary Key:** RentalID

**Foreign Key:** MemberID references Members

**Foreign Key:** BookID references Books

**Relation RentalItems**

RentalID numeric(5)

BookID numeric(5)

RentalDate date

**Primary Key:** RentalItemID, BookID

**Foreign Key:** BookID references Books

# Conclusion

# Appendices

## Appendix A – Title

## Appendix B – Title

Might include:

* **Lookup / Reference tables**
* **Sample reports / Listings**