Software Engineering Project

**DVD Rental System**

Submitted By: **Neil Sebbey (T00198203)**

Computing with **Software Development**

Date Submitted: **09/05/2021**

**Table of Contents**

[1. Introduction/overview 4](#_Toc71495384)

[2. Functional Components 4](#_Toc71495385)

[3. User Requirements 5](#_Toc71495386)

[3.1. RentalSYS will manage members 5](#_Toc71495387)

[3.2. RentalSYS will manage movies 5](#_Toc71495388)

[3.3. RentalSYS will manage rentals 5](#_Toc71495389)

[3.4. RentalSYS will perform administrative reporting 5](#_Toc71495390)

[4. System Requirements 6](#_Toc71495391)

[4.1. System Level Use Case Diagram 6](#_Toc71495392)

[4.2. Manage Members 7](#_Toc71495393)

[4.2.1. Register Member 7](#_Toc71495394)

[4.2.2. Change Member Details 9](#_Toc71495395)

[4.3. Manage Movies 12](#_Toc71495396)

[4.3.1. Add Movie 12](#_Toc71495397)

[4.3.2. Change Movies 14](#_Toc71495398)

[4.3.3. Delete Movie 16](#_Toc71495399)

[4.3.4. Query Movies 18](#_Toc71495400)

[4.4. Process Rentals 19](#_Toc71495401)

[4.4.1. Rent Movie 19](#_Toc71495402)

[4.4.2. Return Movie(s) 23](#_Toc71495403)

[4.4.3. Query Rentals 25](#_Toc71495404)

[4.5. Manage Admin 27](#_Toc71495405)

[4.5.1. Analyse Yearly Revenue 27](#_Toc71495406)

[4.5.2. Analyse Categories 29](#_Toc71495407)

[5. System Model 31](#_Toc71495408)

[5.1. Level-0 DFD 31](#_Toc71495409)

[5.2. Level-1 DFD 31](#_Toc71495410)

[5.3. Level-2 DFD (Process P1: Manage Members) 32](#_Toc71495411)

[5.4. Level-2 DFD (Process P2: Manage Movies) 32](#_Toc71495412)

[5.5. Level-2 DFD (Process P3: Process Rentals) 33](#_Toc71495413)

[6. Data Model (Class Diagram) 34](#_Toc71495414)

[6.1. Class Diagram 34](#_Toc71495415)

[6.2. Relational Schema 35](#_Toc71495416)

[6.3. Database Schema 35](#_Toc71495417)

[7. Conclusion 38](#_Toc71495418)

[8. Appendices 39](#_Toc71495419)

[8.1. Appendix A – Relational schema of the "grid" database 39](#_Toc71495420)

# Introduction/overview

**This program allows members of a DVD Rental Club to rent and return movies more efficiently by storing information about the club’s members and movies inventory, as well as logging rentals and returns of movies by members using the Perform Rental component.**

# Functional Components

The DVD Rental System consists of the four functional components, which are the end-user (members), movies, rentals being made or returned and staff (admin). Members can be added, and their details can be modified at any time. New movies can be added to the system and their details (possibly including price) can be changed at any time. Rentals and Returns can be logged into the system, and the system can provide a list of overdue rentals. The rental company’s staff can list new movie releases and analyse revenue.

# User Requirements

## RentalSYS will manage members

* + 1. RentalSYS will register the details of a member
    2. RentalSYS will allow a member details to be changed

## RentalSYS will manage movies

* + 1. RentalSYS will register the details of a movie
    2. RentalSYS will allow a movie details to be changed
    3. RentalSYS will allow a movie details to be deleted
    4. RentalSYS will produce a list of movies stored on the system

## RentalSYS will manage rentals

* + 1. RentalSYS will record details of a rental
    2. RentalSYS will record a return of a movie
    3. RentalSYS will produce a list of rentals logged onto the system

## RentalSYS will perform administrative reporting

* + 1. RentalSYS will produce a yearly revenue analysis
    2. RentalSYS will produce a yearly analysis by movie category within the current year

# System Requirements

The system requirements of the system are presented in this section of the document.

## System Level Use Case Diagram

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

Receptionist

Member

Manager

Member

## Manage Members

This module is used to create and update entries of members of the DVD Rental System. The Register Member function adds the details of a prospective member into the system. The Change Member Details function modifies the existing details for each member stored into the system, including Names (especially people who changed their surnames after marriage), phone numbers or email addresses.

### Register Member

This function, as the name suggests, registers the details of a new member into the rental system. The receptionist is the primary actor who is making use of the system, while a prospective member (another actor) provides their details when signing up for the DVD rental service. The new member’s details, including Name (surnames and forenames), Email Address and Phone Number will be stored into the system.

Receptionist

R<<extends>>

Display Member Error

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add Member | |
| **Use Case Id** | MM01 | |
| **Priority** | 1 | |
| **Source** | Member | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** | Member | |
| **Description** | This function adds a member to the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Receptionist** | **System** |
|  | **Step 1:** The receptionist invokes the Add Member function  **Step 4:** The receptionist enters the required data:   * Surname * Forename * Phone Number * Street * Town * County * Eircode * Email Address   **Step 5**: The receptionist confirms that the member is to be added by clicking on the Add Member button | **Step 2:** The system assigns the next MemberId and loads on UI  **Step 3:** Display UI  **Step 6:** The system validates the data entered:   * All fields must be entered * Surname and Forename must not be numeric * Phone Number must be numeric, <= 14 * Town and County must not be numeric * Email must not already exist   **Step 7:** The system saves the data in the **Members File**:   * MemberId * Surname * Forename * Phone Number * Street * Town * County * Eircode * Email Address   **Step 8:** Display confirmation message  “*<Surname>*, *<Forename>* - Member ID *nnnn* Added”  **Step 9:** Reset the UI |
| **Alternate Scenarios** | **Receptionist** | **System** |
| **Invalid Data Found** |  | **Step 6**: Invalid Data detected  **Step 7**: Display an appropriate error message |
| **Conclusions** | A member has been added to the system | |
| **Post conditions** | The new member can now rent movies | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Change Member Details

This function modifies the existing details of each member stored onto the rental system. The member is the primary actor who is making use of the system. An existing member’s details, including Surname, Forename, Email Address and Phone Number can be modified at any time, particularly when a member has changed their names after marriage, moved to another household, changed their email address or their phone number.

Receptionist

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Change Member Details | |
| **Use Case Id** | MM02 | |
| **Priority** | 2 | |
| **Source** | Member | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** |  | |
| **Description** | This function changes a member’s details | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Receptionist** | **System** |
|  | **Step 1:** The receptionist invokes the Update Members function  **Step 3**: The receptionist enters a member’s surname  **Step 5:** The receptionist selects the member to be updated  **Step 7**: The receptionist updates the required data:   * Surname * Forename * Phone Number * Street * Town * County * Eircode * Email Address   **Step 8**: The receptionist confirms that the member is to be updated | **Step 2**: Display UI  **Step 4**: The system retrieves the MemberId, Surname, Forename, Phone No. and Email Address of members containing the entered surname and load on UI  **Step 6**: It retrieves all details for the selected member from the Members file and display on the UI for editing  **Step 9:** The system validates the data entered:   * All fields must be entered * Surname and Forename must not be numeric * Phone Number must be numeric, <= 14 * Town and County must not be numeric * Eircode must be <=8 * Email must not already exist   **Step 10:** Update the **Members File:**   * Surname * Forename * Phone Number * Street * Town * County * Eircode * Email Address   **Step 12:** Display confirmation message  “*Thank you, <Forename> (<MemberId>).* *Your member details have been updated successfully.*”  **Step 13:** Reset the UI |
| **Alternate Scenarios** | **Receptionist** | **System** |
| **Invalid Data Found** |  | **Step 9**: Invalid Data detected  **Step 10**: Display an appropriate error message |
| **Conclusions** | The member’s personal details have been updated | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Manage Movies

This module is more often used by the manager of the DVD rental service operator to add, update or delete entries of movies, including new movie releases, onto the DVD Rental System. The details for each movie, including Title, Genre and Release Year, are provided by the distributors (including film studios such as Universal Pictures and Disney)

### Add Movie

This function adds one or more movie(s) into the system. The manager is the primary business actor who is responsible for adding movies, especially those that were newly released on DVD, into the Rentals System with the help of the distributor, who will provide the details of the movie (including Title, Genre and Age Rating).

Manager

<<includes>>

Distributor

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add Movie | |
| **Use Case Id** | MV01 | |
| **Priority** | 1 | |
| **Source** | Distributor | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Distributor | |
| **Description** | This function adds a movie to the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invokes Add Movie function  **Step 9:** Enter the required data:   * Title * Genre * AgeRating * Year * Category | **Step 2:** Assign next MovieId  **Step 3**: Set **Year** to today’s system year  **Step 4**: A default **Status** of ‘available’ (**‘A’**) is assigned to the movie  **Step 5**: Retrieve **Genre from Genres file** and load onto UI  **Step 6**: Retrieve **AgeRating from AgeRatings** file and load onto UI  **Step 7**: Retrieve **Category from Categories** file and load onto UI  **Step 8**: Display UI  **Step 10:** The system validates the data entered:   * All fields must be entered/selected * Year must be between 1900 and the current year (2021)   **Step 11:** Save the data in the **Movies File**:   * MovieId * Title * Genre * AgeRating * Year * Category * Status (‘A’)   **Step 12:** Display confirmation message  “*<Title>*, *<Genre>* - Movie Id *<MovieId>* Added”  **Step 13:** Reset the UI |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Found** |  | **Step 10**: Invalid Data detected  **Step 11**: Display an appropriate error message |
| **Conclusions** | The movie has been added to the system | |
| **Post conditions** | This movie can now be rented by members | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Change Movies

This function lets the manager update the details of movies that were already stored on the Rentals System

Manager

<<includes>>

Distributor

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Change Movie | |
| **Use Case Id** | MV02 | |
| **Priority** | 2 | |
| **Source** | Distributor | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Distributor | |
| **Description** | This function updates details of each movie stored on the system. | |
| **Preconditions** |  | |
| **Trigger** | On click from Main Menu | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invokes Change Movie function  **Step 3**: Search for a movie by title (in full or part of)  **Step 5:** Select a movie to be updated  **Step 8**: Change the required data to be updated:   * Title * Genre * AgeRating * Year * Category | **Step 2**: Display UI  **Step 4**: Retrieve and display list of Movie entries by the entered title  **Step 7**: Retrieve all details for selected movie from Movies file and load on UI for editing:   * Title * Genre * AgeRating * Year * Category * Status   **Step 9:** The system validates the data entered:   * Status must not be ‘D’ (Deleted) or ‘U’ (Unavailable) * All fields must not be null   **Step 10:** Update the data in the **Movies File** specifically for one or more fields that are altered  **Step 11:** Display confirmation message  “Movie *<MovieId>* Updated”  **Step 12:** Reset the UI |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Found** |  | **Step 9**: Invalid Data detected  **Step 10**: Display an appropriate error message |
| **Conclusions** | The details for the movie have been updated | |
| **Post conditions** | The updated details will be displayed on other program components | |
| **Business Rules** | The details for a movie cannot be changed when a movie has been rented (i.e. the status is set to ‘U’ (Unavailable)) or Deleted (‘D’) | |
| **Implementation Constraints** |  | |

### Delete Movie

This function removes a movie entry from the Rentals System

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Delete Movie(s) | |
| **Use Case Id** | MV03 | |
| **Priority** | 3 | |
| **Source** | System | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function sets the status to ‘D’ (Deleted) | |
| **Preconditions** |  | |
| **Trigger** | On click from Main Menu | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invokes Delete Movie function  **Step 3**: Manager enters the title (or a part of the title) of the movie to be deleted  **Step 5:** Select a movie to be deleted  **Step 8**: User confirms the deletion of movie | **Step 2:** Display UI  **Step 4**: Retrieve a list of movies with matching title or part of a title and load on UI  **Step 6**: Retrieve Movie Details (MovieId, Title, AgeRating, Year, Category, Status) from Movies file and load on UI  **Step 7**: Display confirmation message: “Are you sure to delete this movie?”  **Step 9:** The system validates the data entered:   * Status must not be ‘D’ (Deleted) or ‘U’ (Unavailable)   **Step 10:** The system alters the data in the **Movies** File:   * Status (**D**eleted)   **Step 11:** Display confirmation message  “Movie *<MovieId>* has been deleted”  **Step 12:** Reset the UI |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Found** |  | **Step 9**: Invalid Data detected  **Step 10**: Display appropriate error message |
| **Conclusions** | A movie has been deleted from the system | |
| **Post conditions** | This movie will no longer be rented by members | |
| **Business Rules** | The movie cannot be deleted when its status was set to ‘U’, i.e. a movie has been rented | |
| **Implementation Constraints** |  | |

### Query Movies

This function provides a list of movies that were entered into the Rentals System, with a text box used to search for a movie by title.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Query Movie(s) | |
| **Use Case Id** | MV04 | |
| **Priority** | 4 | |
| **Source** | System | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | It displays a list of all movies stored on the system, by searching for the title. | |
| **Preconditions** | Each movie entry must be present on the system | |
| **Trigger** | On click from the main menu | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invokes Query Movie function  **Step 3**: Manager enters the title of the movie  **Step 5:** Select a movie to view more details | **Step 2:** Display UI  **Step 4**: Retrieve Movie Details (MovieId, Title, AgeRating, Year, Category, Status) from **Movies** file and load on UI  **Step 6**: Display list of movie details |
| **Alternate Scenarios** | **Manager** | **System** |
|  |  |  |
| **Conclusions** | A list of movie details have been displayed, after searching for the title | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Process Rentals

### Rent Movie

The *Rent Movie* function allows the member to process transactions of renting a movie. This function is primarily used by the member. As the other participating actor, the receptionist can also process transactions on the member’s behalf when the member needs assistance from the rental club’s staff, especially the receptionist.

Receptionist

Member

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Rent Movie(s) | |
| **Use Case Id** | RT01 | |
| **Priority** | 1 | |
| **Source** | Member | |
| **Primary Business Actor** | Member | |
| **Other Participating Actors** | Receptionist | |
| **Description** | This function creates an order of movies available for rent using a checkout system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Member** | **System** |
|  | **Step 1:** Invokes Rent Movie function  **Step 3:** Enter Member ID  **Step 6**: User enters a movie title (or a part of a title)  **Step 8**: Select a movie to be added to the cart  **Step 11**: Click on Rent Movie(s) to confirm transaction | **Step 2:** Display UI  **Step 4:** The system validates the MemberId;   * Must be entered * Must be numeric   **Step 5**: Retrieve member details from **Members** file and display on UI.  **Step 7**: Retrieve Movies available to rent with matching title and load onto UI  **Step 9:** Assign next RentalId  **Step 10**: The movie has been added to cart. Display the checkout on UI  **Step 12**: Save the data in the **Rentals File**:   * RentalId * MemberId * RentDate * DueDate * Cost   **Step 13**: Save the data in the **RentalItems File**:   * RentalId * MovieId * Category   **Step 14:** Display confirmation message  “Thank you for renting movies from us.”  **Step 15:** Reset the UI |
| **Alternate Scenarios** | **Member** | **System** |
| **Select Member by Searching Surname**  **Adding second movie to cart and then add a third (or more) movie to cart**  **Adding another movie and then finish the transaction**  **Invalid Data Found (MemberId)** | **Step 3:** Enter Member’s Surname  **Step 5:** The user selects the member who will rent the movie(s)  **Step 7**: User enters a movie title (or a part of a title)  **Step 9**: Select a movie to be added to the cart  **Step 11**: Either add another movie or confirm transaction  **Step 11**: Select another movie to be added to the cart  **Step 13**: The user can add another movie by repeating step 11  **Step 11**: Select another movie to be added to the cart  **Step 13**: Click on Rent Movie(s) to confirm transaction | **Step 4**: Retrieve and display list of Member entries by the entered surname  **Step 6**: Retrieves all details for the selected member from the Members file and display on the UI for editing  **Step 8**: Retrieve Movies available to rent with matching title and load onto UI  **Step 10:** Assign next RentalId  **Step 12**: Another movie has been added to cart.  **Step 12**: Another movie has been added to cart.  **Step 14**: Save the data in the **Rentals File**:   * RentalId * MemberId * RentDate * RentTime * Cost   **Step 15**: Save the data in the **Rentals File**:   * RentalId * MovieId * Category   **Step 16:** Display confirmation message  “Thank you for renting movies from us.”  **Step 17:** Reset the UI  **Step 4**: Invalid Data detected  **Step 5**: Display an appropriate error message |
| **Conclusions** | The system logs a member’s rental of movies | |
| **Post conditions** | The member has rented movies | |
| **Business Rules** | The movie status must not be Unavailable (‘U’) or Deleted (‘D’) to ensure that the member only rents a movie that is available to rent | |
| **Implementation Constraints** |  | |

### Return Movie(s)

The *Return Movie(s)* function allows the Member or the Receptionist to put the movies back onto the Rentals System by selecting the checkbox for movies that the member has rented.

Member

<<includes>>

Receptionist

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Return Movie | |
| **Use Case Id** | RT01 | |
| **Priority** | 1 | |
| **Source** | Member | |
| **Primary Business Actor** | Member | |
| **Other Participating Actors** | Receptionist | |
| **Description** | This function logs the return of a movie to the rental club | |
| **Preconditions** |  | |
| **Trigger** | On click from Main Menu | |
| **Expected Scenario** | **Member** | **System** |
|  | **Step 1:** Invokes Return Movie(s) function  **Step 3:** Enter the Rental ID  **Step 5**: Member selects their Rental ID  **Step 8**: Member enters the Movie ID  **Step 10**: Member clicks the entry on the list. Click **Return Movie** and then click ***Yes*** to confirm | **Step 2:** Display UI  **Step 4**: System validates the **Rentalid**;   * Must be entered * Must be numeric   **Step 6**: Retrieve Rental details from **Rentals** file and load on UI  **Step 7**: Retrieve Member details from **Members** file based on the Member ID data within the MemberId column of **Rentals** file  **Step 9**: Retrieve RentalId, MovieId, Category from **RentalItems** file; MemberId from **Rentals** file; and Title, Status from **Movies** file, with matching MovieId and MemberId and load on UI  **Step 10**: Alter the **RentalItems** table and save the following data:   * **ReturnedDate** (System.DateTime.Now) * **ReturnedByMemId**   **Step 7**: Set status of movie to ‘**A**’ (Available) in **Movies** table  **Step 8:** Display confirmation message  “*Thank you for returning movies to us, <Forename>.*”  **Step 9:** Reset the UI |
| **Alternate Scenarios** | **Member** | **System** |
| **Invalid Rental ID Found** |  | **Step 4**: Invalid Rental ID Data detected  **Step 5**: Display an appropriate error message |
| **Conclusions** | The system sets the availability status of the movie back to ‘A’ | |
| **Post conditions** | The next member can now rent the movie | |
| **Business Rules** | The member must bring the physical discs for return | |
| **Implementation Constraints** |  | |

### 4.4.3. Query Rentals

The Query Rentals function displays a detailed information about a rental by entering the member’s rental ID and it will display a list of Movie items listed within the RentalItems table.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Query Rentals | |
| **Use Case Id** | RT04 | |
| **Priority** | 4 | |
| **Source** | System | |
| **Primary Business Actor** | Member | |
| **Other Participating Actors** | Receptionist | |
| **Description** | This function displays a list of rentals being made. | |
| **Preconditions** |  | |
| **Trigger** | On click from Main Menu | |
| **Expected Scenario** | **Member** | **System** |
|  | **Step 1:** Invokes Return Movie(s) function  **Step 3:** Enter the Rental ID  **Step 5**: Member selects their Rental ID  **Step 7**: Member clicks on View Your Rentals button.  **Step 9**: Member clicks on each movie entry on their Rented Movies list. | **Step 2:** Display UI  **Step 4**: System validates the **Rentalid**;   * Must be entered * Must be numeric   **Step 6**: Retrieve Rental details from **Rentals** file and load on UI  **Step 8**: Retrieve RentalId, MovieId, Category from **RentalItems** file; MemberId from **Rentals** file; and Title, Status from **Movies** file, with matching RentalId and MemberId and load on UI  **Step 10**: Retrieve data from Movies file, based on the selected cell on their Rented Movies list and load on UI |
| **Alternate Scenarios** | **Member** | **System** |
| **Invalid Rental ID found** |  | **Step 4**: Invalid Rental ID Data detected  **Step 5**: Display an appropriate error message |
| **Conclusions** | The system displays an output of all rentals being made | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Manage Admin

This component is more often used by the manager of a movie rental club and it performs administrative reporting of business operations, including yearly revenue and overall report of transactions by category.

### Analyse Yearly Revenue

The Analyse Yearly Revenue function calculates the total yearly revenue.

Manager

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Analyse Yearly Revenue | |
| **Use Case Id** | AD01 | |
| **Priority** | 1 | |
| **Source** | System | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function displays a list of movies that were newly released for the current month. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invokes Analyse Yearly Revenue function  **Step 3:** Select a year to be displayed  **Step 7**: When finished, close the report window | **Step 2:** Display UI  **Step 4:** Retrieve Transactions data from **Rentals** file and load onto UI  **Step 5:** Calculate the total revenue for specific year by selecting rental entries from the RentDate column of the **Rentals** file containing the Year from input  **Step 6:** Display the total yearly revenue report  **Step 8:** Close UI |
| **Alternate Scenarios** | **Manager** | **System** |
|  |  |  |
| **Conclusions** | A bar chart diagram report of annual revenue has been displayed | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Analyse Categories

The Analyse Categories function produces an administrative report of rental categories. This allows the manager to calculate the total revenue of each rental category within the current year, i.e. New Releases or Best Selling by week, month or year.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Analyse Categories(s) | |
| **Use Case Id** | AD02 | |
| **Priority** | 2 | |
| **Source** | System | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function analyses the rental revenue for each rental category within the current year, i.e. New Releases (NR) or Old Releases (OR) in 2021 | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invokes Analyse Categories function  **Step 3:** Select a category to be displayed  **Step 7**: When finished, close the report window | **Step 2:** Display UI  **Step 4:** Retrieve Transactions data from **Rentals** file and load onto UI  **Step 5:** Calculate the total revenue for specific category by selecting rental entries from the Category column of the RentalItems table from input  **Step 6:** Display the total report for specific category within the current year  **Step 8:** Close UI |
| **Alternate Scenarios** | **Manager** | **System** |
|  |  |  |
| **Conclusions** | The total revenue for the selected category within the present year is displayed on screen | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

RentalSYS

Rental Details

DVD

Member

## Level-1 DFD

Member Details

Member

Member Confirmation

Member Details

Process Rentals

P3

Manage Members

P1

Member Details

Members File

Rental Details

Rental Details

Movie Details

Movie Details

Movies File

Process Admin

P4

Manage Movies

P2

Rentals File

RentalItems File

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

Change Member

Members File

Add Member

Member Details

Member Details

Member Details

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

Add Movie

Change Movie

Movie Details

Movie Details

Movie Details

Movies File

Delete Movies

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

RentalItems File

Rentals File

Process Rentals

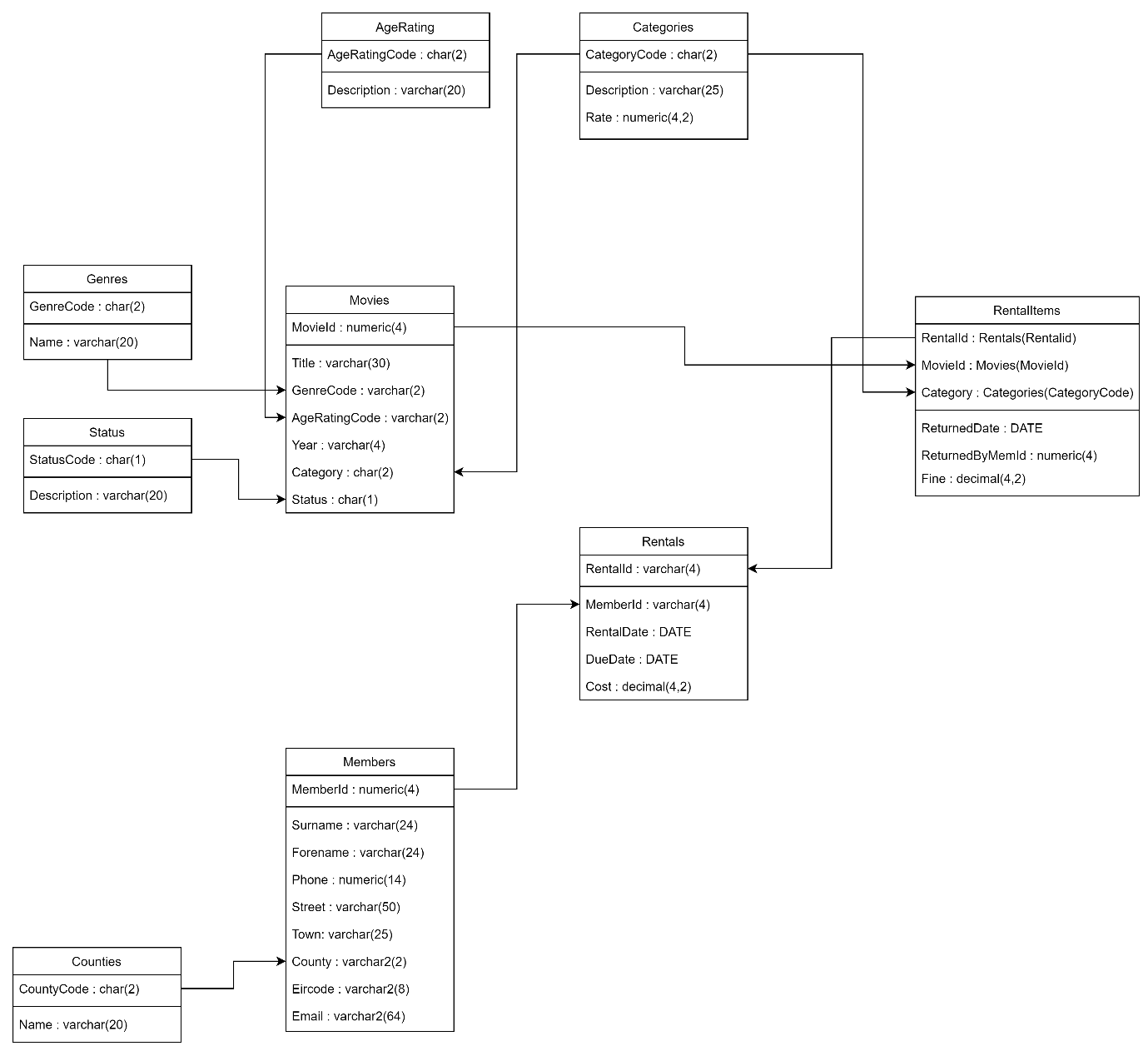
Movies File

Members File

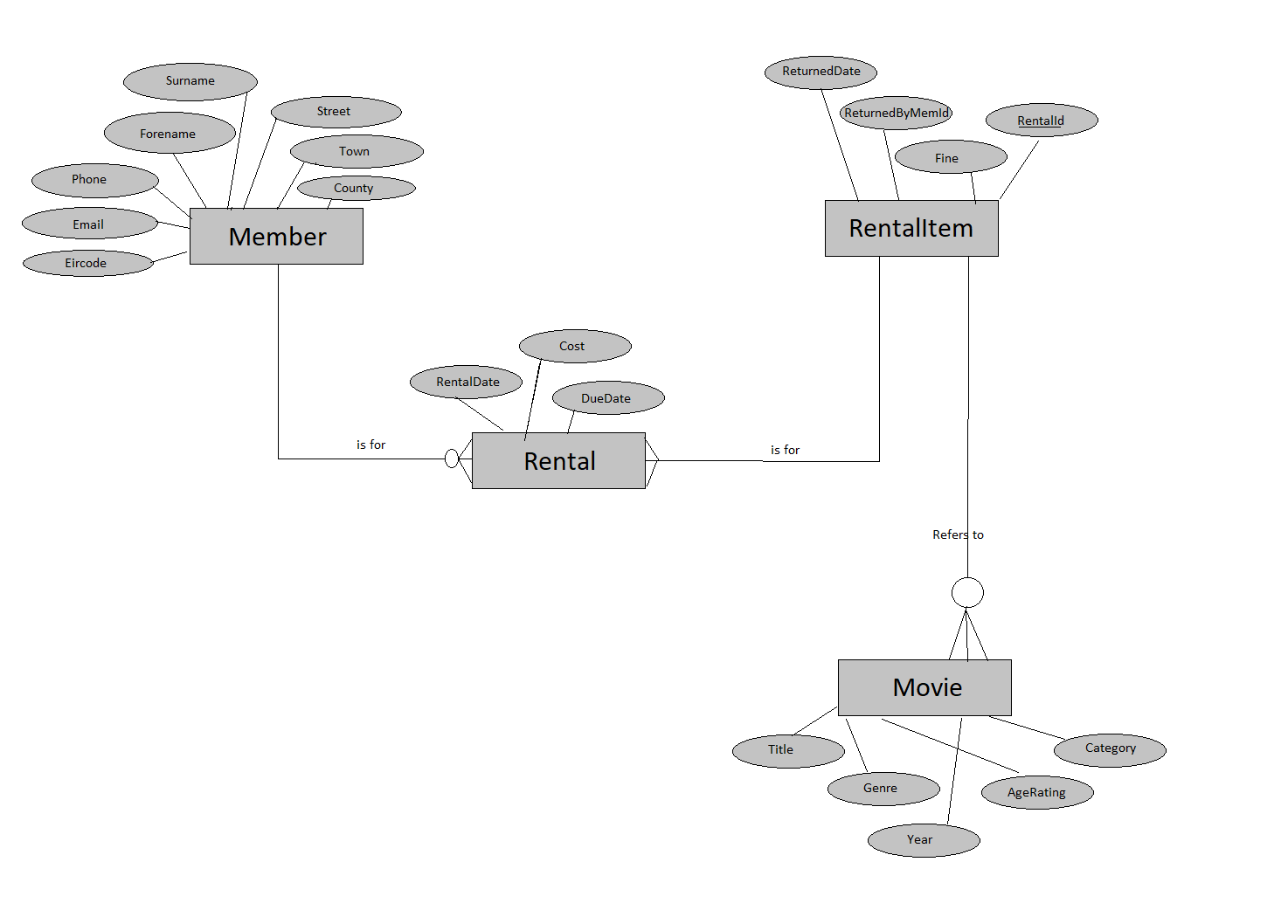
# Data Model (Class Diagram)

The DVD Rentals System describes the data types and attributes using Class Diagrams and Schemas. The fields that end with ‘-Id’ are usually numeric data with 4 characters maximum, while the other fields are usually varchar data, except date and time which they have their own respective data types.

## Class Diagram



## Relational Schema



## Database Schema

**Relation**: Counties

Attributes:

* CountyCode char(2)
* Name varchar(20),

**Primary** Key: CountyCode

**Foreign** Key:

**Relation**: Genres

Attributes:

* GenreCode char(2)
* Name varchar(20),

**Primary** Key: GenreCode

**Foreign** Key:

**Relation**: AgeRating

Attributes:

* AgeRatingCode char(2)
* Description varchar(20),

**Primary** Key: AgeRatingCode

**Foreign** Key:

**Relation**: Categories

Attributes:

* CategoryCode char(2)
* Description varchar(25),
* Rate numeric(4,2)

**Primary** Key: CategoryCode

**Foreign** Key:

**Relation**: Status

Attributes:

* StatusCode char(1)
* Description varchar(20),

**Primary** Key: StatusCode

**Foreign** Key:

**Relation**: Members

Attributes:

* MemberId numeric(4)  NOT NULL
* Surname varchar2(24), NOT NULL
* Forename varchar2(24), NOT NULL
* Phone numeric (14), NOT NULL
* Street varchar2(50), NOT NULL
* Town varchar2(25), NOT NULL
* County varchar2(2), NOT NULL
* Eircode varchar2(8), NOT NULL
* Email varchar2(64), NOT NULL

**Primary** Key: MemberId

**Foreign** Key:

**Relation**: Movies

Attributes:

* MovieId numeric(4)  NOT NULL
* Title varchar(30), NOT NULL
* GenreCode varchar(2), NOT NULL
* AgeRatingCode char (2), NOT NULL
* Year numeric(4), NOT NULL
* Category char(2), NOT NULL
* Status char(1), NOT NULL

**Primary** Key: MovieId

**Foreign** Key: Genre references Genres(GenreCode)

AgeRatingCode references AgeRating(AgeRatingCode)

Category references Categories(CategoryCode)

Status references Status(StatusCode)

**Relation**: Rentals

Attributes:

* RentalId numeric(4)  NOT NULL
* MemberId numeric(4)  NOT NULL
* RentalDate DATE
* DueDate DATE
  + Cost decimal(4,2) NOT NULL

**Primary** Key: RentalId

**Foreign** Key: MemberId references Members

**Relation**: RentalItems

Attributes:

* RentalId NOT NULL REFERENCES Rentals(RentalId) ON DELETE CASCADE
* MovieId NOT NULL REFERENCES Movies(MovieId) ON DELETE CASCADE
* Category NOT NULL REFERENCES Categories(CategoryCode) ON DELETE CASCADE
* ReturnedDate DATE
* ReturnedByMemId numeric(4)
  + Fine decimal(4,2)

**Primary** Key: RentalId, MovieId, Category

**Foreign** Key:

# Conclusion

**The DVD Rentals System has functional components that will help the customers (members) and staff (managers and receptionists) of the DVD Movie Rental Club get the job done. The receptionist, as the primary actor, can add or change details of the customer. The manager is the primary business actor for certain tasks, e.g. adding or changing details of movies. The member has a major role as a primary business actor in processing rentals and manging members, but the receptionist is also involved on the behalf of the member if they need any assistance with the rental service.**

# Appendices

## Appendix A – Relational schema of the "grid" database

