Requirements Engineering

Viter Game Store

Submitted By: Mykhailo Hnylytskyi (T00257938)

Computing with Software Development

Date Submitted: 07/12/2024

Table of Contents

[1. Introduction/overview 4](#_Toc184480191)

[2. Functional Components 5](#_Toc184480192)

[3. User Requirements 5](#_Toc184480193)

[3.1. Game Store will manage Games 5](#_Toc184480194)

[3.2. Game Store will process account creation 5](#_Toc184480195)

[3.3. Game Store will perform salles 6](#_Toc184480196)

[3.4. Game Store will perform administrative reporting 6](#_Toc184480197)

[4. System Requirements 7](#_Toc184480198)

[4.1. System Level Use Case Diagram 7](#_Toc184480199)

[4.2. Manage Games 8](#_Toc184480200)

[4.2.1. Add Category 8](#_Toc184480201)

[4.2.2. Update Category 10](#_Toc184480202)

[4.2.3. Add Game 12](#_Toc184480203)

[4.2.4. Update Game 14](#_Toc184480204)

[4.2.5. Remove Game 16](#_Toc184480205)

[4.3. Account 18](#_Toc184480206)

[4.3.1. Add Account 18](#_Toc184480207)

[4.3.2 Update Account 20](#_Toc184480208)

[4.3.3. Close Account 22](#_Toc184480209)

[4.4. Salles 24](#_Toc184480210)

[4.4.3. Sell Game 24](#_Toc184480211)

[4.4.4. Return Games 27](#_Toc184480212)

[4.5. Admin 29](#_Toc184480213)

[4.5.1. Show Yearly Revenue Analysis 29](#_Toc184480214)

[4.5.2. Show Games Analysis 30](#_Toc184480215)

[5. System Model 33](#_Toc184480216)

[5.1. Level-0 DFD 33](#_Toc184480217)

[5.2. Level-1 DFD 34](#_Toc184480218)

[5.3. Level-2 DFD (Process P1: Process Games) 35](#_Toc184480219)

[5.4. Level-2 DFD (Process P2: Process Accounts) 35](#_Toc184480220)

[5.5. Level-2 DFD (Process P3: Process Salles) 36](#_Toc184480221)

[5.6. Level-2 DFD (Process P4: Process Admin) 37](#_Toc184480222)

[6. Data Model (UML Class Diagram) 39](#_Toc184480223)

[6.1. Class Diagram 39](#_Toc184480224)

[6.2. Relational Schema 40](#_Toc184480225)

[6.3. Database Schema 40](#_Toc184480226)

[7. Conclusion 42](#_Toc184480227)

# Introduction/overview

The document provides a comprehensive analysis and design for the "Viter Game Store" management system, which is aimed at streamlining game retail operations. It outlines the system's purpose, which is to facilitate efficient management of core business functions such as game inventory, user accounts, transactions, and administrative reporting. The system is structured to enhance the user experience for both store managers and customers by automating processes, ensuring data integrity, and enabling robust analysis of sales and performance.

The primary objectives of the system are to manage the addition, update, and removal of games and categories, handle customer account creation and modification, process game sales and returns, and provide detailed analytics for managerial oversight. The document also highlights the importance of designing an intuitive and scalable system that aligns with industry standards, employing data flow diagrams (DFDs) and database schemas to illustrate the system architecture. By addressing the functional and system requirements, the system aims to deliver an optimised digital solution to the challenges of game retail management, thereby enabling growth and improved customer satisfaction.

# Functional Components

This section presents the functional components of the propsoed software system.

# User Requirements

This section describes the user requirements (functional components) as high-level abstract statements.

## Game Store will manage Games

* + 1. **Game Store** will add a game category
    2. **Game Store** will update a game category
    3. **Game Store** will add a game
    4. **Game Store** will update a game
    5. **Game Store** will remove a game

## Game Store will process account creation

* + 1. Game Store will add account
    2. Game Store will close account
    3. Game Store will update account

## 3.3. Game Store will perform salles

3.3.1 Game Store will sell games

* + 1. Game Store will record the return of a game

## 3.4. Game Store will perform administrative reporting

3.4.1. Game Store will produce a yearly revenue analysis

3.4.2. Game Store will produce top sellers games analysis

# System Requirements

This module contains functions to set a category for games and update a category, to add a game(which defines the nightly rate), to update a game and remove game, to add an account(which defines the loyalty points), to update an account and remove an account, to make a salle and return the game, to show yearly revenue analysis and games analysis.

## System Level Use Case Diagram

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

Manager

Guest

Admin

## Manage Games

This module contains functions to add a game category, update a game category, add a game, update a game and remove a game.

### Add Category

This function adds a category that can be used for specified games, for example RC – Race, AC – action, SM - Simulator.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add Category | |
| **Use Case Id** | AC001 | |
| **Priority** | 2 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function creates a game category. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:**  Manager invokes the Add Game Category Function  **Step 3:**  The manager enters the required data:   * Category\_ID (Char 2) * Description (String 20) | **Step 2:** The system displays the UI  **Step 4:** The system validates the data entered:   * All fields must be entered * Category\_ID must be entered * Category\_ID must not be numeric * Category\_ID must not already exist * Description must not be numeric   **Step 5:** The system saves the category details in the Categories File:   * Category\_ID * Description   **Step 6:** The system displays a confirmation message  **Step 7:**  The system resets the UI |
| **Alternate Scenarios** | **Manager** | **System** |
| Invalid Data Entered |  | **Step 4:** Validation test fails  **Step 5:** Display and appropriate error message  **Step 6:** return to Step 3 |
| **Conclusions** | The category has been recorded in the Categories File. | |
| **Post conditions** | This category can now be added. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update Category

This function updates a category to Category File.

|  |
| --- |
| Manager  <<extends>>  <<includes>> |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Category | |
| **Use Case Id** | AC002 | |
| **Priority** | 3 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function updates a category to Categories File. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:**  Manager invokes the Update Game Category Function  **Step 4:**  The manager selects Game Category  **Step 6:** The manager updates required data:   * Category\_ID (char 2) * Description (String 20) | **Step 2:** Retrieve Update Game Category from the Categories File and loads on UI  **Step 3:** The system displays the UI  **Step 5:** The system retrieves all details for Selected Game Category on UI  **Step 7:** The system validates the data entered:   * All fields must be entered * Category\_ID must not be numeric * Category\_ID must not already exist * Description must not be numeric   **Step 8:** The system saves changes to Categories File:   * Category\_ID (char 2) * Description (String 20)   **Step 9:** The system displays a confirmation message  **Step 10:**  The system resets the UI |
| **Alternate Scenarios** | **Manager** | **System** |
| Invalid Data Entered |  | **Step 7:** Validation test fails  **Step 8:** Display and appropriate error message  **Step 9:** return to Step 6 |
| **Conclusions** | The category should be update to the Categories File. | |
| **Post conditions** | Another category may now be added to the Categories File. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Add Game

This function add game to Market File, which keeps track of the game available for purchase.

Store Manager

<<extends>>

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add Game | |
| **Use Case Id** | AG001 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function creates a game. This function determines the category to a game. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:**  Manager invokes the Add Game Function  **Step 5:**  The manager enters the required data:   * Name (String 20) * Description (String 20) * Rate (Double) * Category\_ID (char 2) | **Step 2:** The system assigns a Game\_ID value  **Step 3:** The system retrieve data from a Category File to be displayed on UI  **Step 4:** The system displays the UI  **Step 6:** The system validates the data entered:   * All fields must be entered * Name must not already exist * Name must not be numeric * Category\_ID must be selected * Description must not be numeric * Rate must be numeric(decimal) and greater than zero   **Step 7:** Status is set to ‘Available’ (‘A’)  **Step 8:** The system saves the game details in the Games File:   * Game\_ID * Name * Description * Category\_ID * Rate * Status   **Step 9:** The system displays a confirmation message  **Step 10:**  The system resets the UI |
| **Alternate Scenarios** | **Manager** | **System** |
| Invalid Data Entered |  | **Step 6:** Validation test fails  **Step 5:** Display and appropriate error message  **Step 6:** return to Step 5 |
| **Conclusions** | The game has been recorded in the Games File. | |
| **Post conditions** | Game now has Status Available by default. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### 4.2.4. Update Game

This function update details for game to database Games File.

Store Manager

<<extends>>

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Game | |
| **Use Case Id** | AG002 | |
| **Priority** | 2 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function updates a game. This function change game details. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:**  Manager invokes the Update Game Function  **Step 4:**  The manager enters Game\_ID to find specific game to update  **Step 6:** The manager updates required data:   * Name (String 15) * Category (char 2) * Description (String 20) * Rate (Double) | **Step 2:** Retrieve data from Games File to be displayed on UI  **Step 3:** The system displays the UI  **Step 5:** The system retrieves all details for entered Game from Games File on UI  **Step 7:** The system validates the data entered:   * All fields must be entered * Name must not already exist * Name must not be numeric * Category must be selected * Description must not be numeric * Rate must be numeric(decimal) and greater than zero   **Step 8:** Status is set to ‘Available’ (‘A’)  **Step 9:** The system saves changes to Games File:   * Game\_ID * Name * Description * Rate * Status   **Step 10:** The system displays a confirmation message  **Step 11:**  The system resets the UI |
| **Alternate Scenarios** | **Manager** | **System** |
| Invalid Data Entered |  | **Step 7:** Validation test fails  **Step 8:** Display and appropriate error message  **Step 9:** return to Step 6 |
| **Conclusions** | The game has been updated in the Games File. | |
| **Post conditions** | Updated Game may now be added to the Games File. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### 4.2.5. Remove Game

This function remove game from database File Market.

Store Manager

<<extends>>

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Remove Game | |
| **Use Case Id** | AG003 | |
| **Priority** | 3 | |
| **Source** | Manager | |
| **Primary Business Actor** | Store Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function removes a game from Games File. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:**  Manager invokes the Update Game Function  **Step 4:**  The manager enters Game\_ID to find specific game to update  **Step 7:** The manager updates required data:   * Status (char 1) | **Step 2:** Retrieve data from Games File to be displayed on UI  **Step 3:** The system displays the UI  **Step 5:** The system validates the data entered:   * Game\_ID   **Step 6:** The system retrieves all details for entered Game from Game File on UI  **Step 8:**  The system validates the data entered:   * Status set to “U”   **Step 9:** The system saves changes to Games File:   * Game\_ID * Name * Description * Rate * Status   **Step 9:** The system displays a confirmation message  **Step 10:**  The system resets the UI |
| **Alternate Scenarios** | **Manager** | **System** |
| Invalid Data Entered  Invalid Data Entered |  | **Step 5:**  Validation test fails  **Step 6:** Not display next step  **Step 7:** Wait for entered specific ID  **Step 8:** Validation test fails  **Step 9:** Display and appropriate error message  **Step 10:** return to Step 7 |
| **Conclusions** | The Status has been changed to Unavailable in the Games File. | |
| **Post conditions** | Another Game may now be added to the Games File. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Account

This module contains functions for managing and processing account.

### Add Account

This function add account to database File Accounts.

Guest

<<extends>>

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add Account | |
| **Use Case Id** | AA001 | |
| **Priority** | 1 | |
| **Source** | Guest | |
| **Primary Business Actor** | Guest | |
| **Other Participating Actors** |  | |
| **Description** | This function adds an account to Accounts File. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Guest** | **System** |
|  | **Step 1:**  Guest invokes the Add Account Function  **Step 3:**  The guest enters the required data:   * First\_Name (String 20) * Last\_Name (String 20) * DOB (Data) * Phone\_No (Numeric) * Email (String 50) | **Step 2:** The system displays the UI  **Step 4:** The system validates the data entered:   * All fields must be entered * First\_Name must not be numeric * First\_Name must be entered * Last\_Name must not be numeric * Last\_Name must be entered * DOB must be entered * Phone\_No must be entered * Phone\_No must be numeric * Email must be entered * Email must contains symbol “@”   **Step 5:** The system saves the account details in the Accounts File:   * Account\_No * First\_Name * Last\_Name * DOB * Phone\_No * Email * Status * Loyalty\_Points   **Step 6:** The system displays a confirmation message  **Step 7:**  The system resets the UI |
| **Alternate Scenarios** | **Guest** | **System** |
| Invalid Data Entered |  | **Step 4:** Validation test fails  **Step 5:** Display and appropriate error message  **Step 6:** return to Step 3 |
| **Conclusions** | The account has been recorded in the Accounts File. | |
| **Post conditions** | Account may now be added to the Accounts File. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update Account

This function finding added account and updates details if needed.

Admin

<<extends>>

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Account | |
| **Use Case Id** | AA002 | |
| **Priority** | 1 | |
| **Source** | Guest | |
| **Primary Business Actor** | Guest | |
| **Other Participating Actors** |  | |
| **Description** | This function adds an account to database. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Guest** | **System** |
|  | **Step 1:**  Guest invokes the Update Account Function  **Step 4:**  The guest enters the required data:   * First\_Name (String 20) * Last\_Name (String 20) * DOB (Data) * Phone\_No (Numeric) * Email (String 50) | **Step 2:** Retrieve Update Account from the Accounts File  **Step 3:** The system displays the UI  **Step 5:** The system validates the data entered:   * All fields must be entered * First\_Name must not be numeric * First\_Name must be entered * Last\_Name must not be numeric * Last\_Name must be entered * DOB must be entered * Phone\_No must be entered * Phone\_No must be numeric * Email must be entered   **Step 6:** The system retrieves all details for Account on UI  **Step 7:** The system saves changes to Accounts File  **Step 8:** The system displays a confirmation message  **Step 9:**  The system resets the UI |
| **Alternate Scenarios** | **Guest** | **System** |
| Invalid Data Entered |  | **Step 5:** Validation test fails  **Step 6:** Display and appropriate error message  **Step 7:** return to Step 3 |
| **Conclusions** | The account should be update to the Accounts File. | |
| **Post conditions** | Another account may now be added to the Accounts File. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Close Account

This function for guest requests about closing account, admin select account that request closing, then account removed from File Accounts and File Close Accounts.

Guest

Admin

<<extends>>

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Close Account | |
| **Use Case Id** | AA003 | |
| **Priority** | 2 | |
| **Source** | Admin | |
| **Primary Business Actor** | Guest | |
| **Other Participating Actors** |  | |
| **Description** | This function closes an account. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **GuestG** | **System** |
|  | **Step 1:**  Admin invokes the Close Account Function  **Step 4:** The admin selects Account  **Step 6:**  The admin enters the required data:   * First\_Name (String 20) * Last\_Name (String 20) * DOB (Data) * Phone\_No (Numeric) * Email (String 50) * Status (char 1) | **Step 2:** Retrieve Remove Account from the Accounts File  **Step 3:** The system displays the UI  **Step 5:** The system retrieves all details for selected Account on UI  **Step 7:** The system validates the data entered:   * All fields must be entered * First\_Name must not be numeric * First\_Name must be entered * Last\_Name must not be numeric * Last\_Name must be entered * Phone\_No must be entered * Phone\_No must be numeric * DOB must be entered * Email must be entered   **Step 8:** The system changes Status the Account from Available to Close to the Accounts File  **Step 9:** The system displays a confirmation message  **Step 10:**  The system resets the UI |
| **Alternate Scenarios** | **Guest** | **System** |
| Invalid Data Entered |  | **Step 4:** Validation test fails  **Step 5:** Display and appropriate error message  **Step 6:** return to Step 3 |
| **Conclusions** | The Status has been changed to Close in the Accounts File. | |
| **Post conditions** | Another Account may now be added to the Accounts File. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Salles

This module contains functions Salle Games to Accounts and Return Games from accounts.

### Sell Game

This function Sell Game from Market File.

Guest

<<extends>>

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Sell Games | |
| **Use Case Id** | SG001 | |
| **Priority** | 1 | |
| **Source** | Guest | |
| **Primary Business Actor** | Guest | |
| **Other Participating Actors** |  | |
| **Description** | This function sells games to the Guest from Game File. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Guest** | **System** |
|  | **Step 1:**  Guest invokes the Sell Game Function  **Step 4:**  The guest filtered games by category or by popularity  **Step 9:** The guest enters required details:   * Card Holder Name (String 60) * Card Number (Numeric 16) * Card Expires (Data) * CVV (Numeric 3) | **Step 2:** Retrieve data from Game File to be displayed on UI  **Step 3:** The system displays the UI  **Step 5:** The system validates the data selected  **Step 6:** The system adds the game selected to cart:   * Game\_ID * Name * Rate   **Step 7:** The system displays an asking message for more games  **Step 8:**  The system displays the UI asking for enter purchase details  **Step 10:** The system validates the data selected:   * All fields must be entered * Card Holder Name must be entered * Card Holder Name must not be numeric * Card Number must be entered * Card Number must be numeric * Card Expires must be selected * CVV must be entered * CVV must be numeric * CVV must be 3 digits   **Step 11:** Add individual Loyalty Points amount for each game buying  **Step 12:** The system adds Loyalty Points to Account File and save its changes  **Step 13:** The system adds date for purchase:   * Purchase\_Date   **Step 14:** The system saves purchase to Salles File  **Step 15:** The system displays a confirmation message  **Step 16:** The system reset the UI  **Step 17:** Exit to Main Menu |
| **Alternate Scenarios** | **Guest** | **System** |
| Invalid Data Entered  Asking for more select more game  Invalid Data Entered |  | **Step 5:** Validation test fails  **Step 6:** Display and appropriate error message  **Step 7:** Return to Step 4  **Step 7:** Answer ‘No’  **Step 8:** Next step 8  **Step 10:**  Validation test fails  **Step 11:** Display and appropriate error message  **Step 12:** Return to Step 9 |
| **Conclusions** | The salles have been recorded in the File Salles. | |
| **Post conditions** | UI exits to Main Menu after confirmation of complete transaction. | |
| **Business Rules** | Add Loyalty Points to Account File. | |
| **Implementation Constraints** |  | |

Guest System

Displays UI

Invoke Sell Game

Selects a Category and a Game

More Game?

Presses the button

[Y]

[N]

Displays UI

Enter Required Data

Validate Data

Valid?

[N]

Error Message

[Y]

Add Loyalty Points and purchase date to the account for each game bought, then save the changes

Save Data in Salles File DB

Display Confirmation

Reset UI

### Return Games

Guest

Admin

<<extends>>

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Return Games | |
| **Use Case Id** | SG002 | |
| **Priority** | 2 | |
| **Source** | Admin | |
| **Primary Business Actor** | Guest | |
| **Other Participating Actors** |  | |
| **Description** | This function request return games. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Guest** | **System** |
|  | **Step 1:**  Guest invokes the Return Games Function  **Step 3:**  The guest enters the required data:   * First\_Name (String 20) * Last\_Name (String 20) * Phone\_No (Numeric) * Email (String 30) * Purchase Data (Data) | **Step 2:**  Retrieve data from Salles File to be displayed on UI  **Step 3:** The system displays the UI  **Step 4:** The system validates the data entered:   * All fields must be entered * First\_Name must not be numeric * First\_Name must be entered * Last\_Name must not be numeric * Last\_Name must be entered * Phone\_No must be entered * Phone\_No must be numeric * Email must be entered * Purchase Data must be selected   **Step 5:** The system saves the request in the Return Games File:   * Account\_No * First\_Name * Last\_Name * Phone\_No * Email * Purchase Data   **Step 6:** The system displays a confirmation message  **Step 7:**  The system resets the UI |
| **Alternate Scenarios** | **Guest** | **System** |
| Invalid Data Entered |  | **Step 4:** Validation test fails  **Step 5:** Display and appropriate error message  **Step 6:** return to Step 3 |
| **Conclusions** | Request may now be added to the Return Games File. | |
| **Post conditions** | The account has been displayed appropriate message in the UI. Exit to Main Menu. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## 4.5. Admin

The admin module includes Manager and Admin who interact with File Salles Games, they can sort it for Analysis.

### 4.5.1. Show Yearly Revenue Analysis

Admin

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Show Yearly Revenue Analysis | |
| **Use Case Id** | SA001 | |
| **Priority** | 1 | |
| **Source** | Admin | |
| **Primary Business Actor** | Admin, Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function displays Yearly Revenue Analysis. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Admin** | **System** |
|  | **Step 1:**  The Admin invokes the Show Yearly Revenue Analysis Function  **Step 4:** The Admin/Manager selects year  **Step 6:** Press back  **Step 7:** Press exit | **Step 2:**  Retrieve data from Salles File to be displayed on UI  **Step 3:** The system displays the UI  **Step 5:** The system counts all salles games, all purchase and retrieve the data:   * Year * Purchase   **Step 6:**  The system resets the UI  **Step 8:** Exit to main Menu |
| **Alternate Scenarios** | **Admin** | **System** |
|  |  |  |
| **Conclusions** | The system displays the Salles File. | |
| **Post conditions** | Exit to main Menu. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### 4.5.2. Show Games Analysis

This function displays all Games and sort it by popularity among Guests.

Admin

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Show Games Analysis | |
| **Use Case Id** | SA002 | |
| **Priority** | 1 | |
| **Source** | Admin | |
| **Primary Business Actor** | Admin | |
| **Other Participating Actors** |  | |
| **Description** | This function displays Games Analysis. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Admin** | **System** |
|  | **Step 1:**  The Admin invokes the Games Analysis Function  **Step 5:** Press back  **Step 7:** Press exit | **Step 2:**  Retrieve data from Salles File to be displayed on UI  **Step 3:** The system sorts the data by quantities of salles games  **Step 4:** The system displays the UI  **Step 6:**  The system resets the UI  **Step 8:** Exit to main Menu |
| **Alternate Scenarios** | **Admin** | **System** |
| Invalid Data Entered |  |  |
| **Conclusions** | The system displays the Salles File and Games File counted by quantities. | |
| **Post conditions** | Exit to main Menu | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

# System Model

The following dataflow diagrams have been produced for the system:

External entities:

Guest

Data Stores:

D1 Categories File

D2 Games File

D3 Accounts File

D4 Salles File

D5 Return Games File

Processes:

P1 Process Games

P1.1 New Category

P1.2 Amend Category

P1.3 New Game

P1.4 Amend Game

P1.5 Remove Game

P2 Process Accounts

P2.1 New Account

P2.2 Close Account

P2.3 Amend Account

P3 Process Salles

P3.1 Sell Games

P3.2 Return Games

P4 Process Admins

P4.1 Issue Yearly Revenue Analysis

P4.2 Issue Games Analysis

## Level-0 DFD

Purchase details

Guest

Game Store System

Purchase Confirmation details

## Level-1 DFD

Category Details

P1

P

Process Games

Account Details

D1

Categories File

P2

Process Accounts

D3

Account File

Category Details

Game Details

Account Details

Account Details

Game Details

D2

Games File

Game Details

Guest

Game Details

Salle Details

Account Details

Salle Details

P4

Process Admin

D4

Salles File

P3

Process Salles

Salle Details

Account Details

D5

Return Games File

Account Details

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

Category Details

Category Details

P1.2

P

Update Category

P1.1

P

Add Category

D1

Categories File

Category Details

Category Details

Game Details

Game Details

Game Details

D2

Games File

Game Details

Game Details

Game Details

Game Details

P1.5

P

Remove Game

P1.3

P

Add Game

P1.4

P

Update Game

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

P2.1

P

Add Account

Account Details

P2.2

P

Close Account

Account Details

D3

Account File

P2.3

P

Update Account

Account Details

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

P3.1

P

Sell Games

D4

Salles File

Sell Details

P3.2

P

Return Games

D5

Return Games File

Return Detail

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

P4.1

4

P

Issue Yearly Revenue Analysis

Salle Details

Salle Details

P4.2

4

P

Issue Game Analysis

Game Details

D2

Games File

D4

Salles File

# Data Model (UML Class Diagram)

Data Model represents the structure of the “Viter Game Store” management system. It organises operations and entities involved in the management of games, accounts, salles, and administrative tasks.

## Class Diagram

Games

-Game\_ID\*: Numeric

-Name: String (20)

-Description: String (20)

-Category: String

-Rate: double

-Status: Char (1)

Salles

-Salle\_ID\*: Numeric

-Game\_ID: Numeric

-Account\_ID: Numeric

-Date\_Purchase: DATE

-Amount: double

à

0..\*

1..\*

for

0..\*

1..\*

1..\*

make

has a

à

à

1..\*

0..\*

Accounts

-Account\_No\*: Numeric

-First\_Name: String (20)

-Last\_Name: String (20)

-DOB: DATE

-Phone\_No: Numeric

-Email: String (50)

-Status: String (1)

-Loyalty\_Points: Numeric

Category

-Category\_ID\*: Char (2)

-Description: Char (20)

for

à

1..\*

Admin

-Report\_ID\*: Numeric

-Report\_Type: String

-Salle\_ID: Numeric

-Game\_ID: Numeric

-Report\_Generated\_Date: DATE

## Relational Schema

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

*Category* (Category\_ID\*, Description)

*Games* (Game\_ID\*, Name, Description, Rate, Status)

*Salles* (Sale\_ID\*, Game\_ID, Account\_ID, Date\_Purchase, Amount)

*Accounts* (Account\_ID\*, FirstName, LastName, DOB, Phone\_No, Email, Status, Loyalty\_Points)

*Admin* (Report\_ID\*, Report\_Type, Report\_Generated\_Date)

## Database Schema

Schema: Viter Game Store

Relation: Category

Attributes: Category\_ID Char (2),

Description: String (20)

Primary Key: Category\_ID

Relation: Games

Attributes: Game\_ID: Numeric,

Name: String (20),

Description: String (20),

Category\_ID: Char (2),

Rate: double NOT NULL,

Status: Char (1)

Primary Key: Game\_ID

Foreign Key: (Category\_ID) references Category (Category\_ID)

Relation: Salles

Attributes: Salle\_ID: Integer,

Game\_ID: Numeric,

Account\_ID: Numeric,

Date\_Purchase: DATE,

Amount: double NOT NULL

Primary Key: Salle\_ID

Foreign Key: (Game\_ID) references Games (Game\_ID)

Foreign Key: (Account\_ID) references Accounts (Account\_ID)

Relation: Accounts  
 Attributes: AccountNo: Numeric,

FirstName: String (20) NOT NULL,

LastName: String (20) NOT NULL,

DOB: DATE,

PhoneNo: Numeric NOT NULL,

Email: String (50) NOT NULL UNIQUE,

Status: Char (1),

LoyaltyPoints: Numeric

Primary Key: AccountID

Relation: Admin

Attribute: Report\_ID\*: Numeric,

Report\_Type: String (20),

Salle\_ID: Numeric,

Game\_ID: Numeric,

Report\_Generated\_Date: DATE

Primary Key: Report\_ID

Foreign Key: (GameID) references Games (GameID)

# Conclusion

The analysis and design of the Viter Game Store management system underscore its potential to transform game retail operations through process automation and data-driven decision-making. The system's architecture is designed to efficiently manage game inventories, customer accounts, and transactional data while providing actionable insights through administrative reporting features. The implementation of a well-structured relational database schema ensures data consistency and supports the scalability of the system.

The findings emphasise the significance of user-centric design in achieving operational efficiency and enhancing the customer experience. Furthermore, the proposed system offers a foundation for future enhancements, such as integrating advanced analytics or expanding functionality to support multiple branches. Management system represents a strategic tool for modernising business operations and driving competitive advantage in the dynamic gaming industry.