Requirements Engineering

Linen Order SYS

Submitted By: Michael Edgar – T00194492

Computing with Games Development

Date Submitted: dd/mm/yyyy

**Table of Contents**

[1. Introduction/overview 3](#_Toc474162299)

[2. Functional Components 4](#_Toc474162300)

[3. User Requirements 5](#_Toc474162301)

[4. System Requirements 6](#_Toc474162302)

[4.1. System Level Use Case Diagram 6](#_Toc474162303)

[4.2. Module 1 (Use active Verb + Noun) 6](#_Toc474162304)

[4.2.1. Functional Requirement 1 (Use active verb + noun) 6](#_Toc474162305)

[4.2.2. Functional Requirement 2 (Use active verb + noun) 6](#_Toc474162306)

[4.2.3. Functional Requirement 3 (Use active verb + noun) 6](#_Toc474162307)

[4.3. Module 2 (Use active Verb + Noun) 6](#_Toc474162308)

[4.3.1. Functional Requirement 1 (Use active verb + noun) 6](#_Toc474162309)

[4.3.2. Functional Requirement 2 (Use active verb + noun) 6](#_Toc474162310)

[4.4. Module 3 (Use active Verb + Noun) 6](#_Toc474162311)

[4.4.1. Functional Requirement 1 (Use active verb + noun) 6](#_Toc474162312)

[4.4.2. Functional Requirement 2 (Use active verb + noun) 6](#_Toc474162313)

[5. System Model 7](#_Toc474162314)

[5.1. Level-0 DFD 7](#_Toc474162315)

[5.2. Level-1 DFD 7](#_Toc474162316)

[5.3. Level-2 DFD (Process P1: Title) 7](#_Toc474162317)

[5.4. Level-2 DFD (Process P2: Title) 7](#_Toc474162318)

[5.5. Level-2 DFD (Process P3: Title) 7](#_Toc474162319)

[6. Data Model (Class Diagram) 8](#_Toc474162320)

[6.1. Class Diagram 8](#_Toc474162321)

[6.2. Relational Schema 8](#_Toc474162322)

[6.3. Database Schema 8](#_Toc474162323)

[7. Conclusion 9](#_Toc474162324)

[8. Appendices 10](#_Toc474162325)

[8.1. Appendix A – Title 10](#_Toc474162326)

[8.2. Appendix B – Title 10](#_Toc474162327)

# Introduction/overview

# 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.

* **3.1** Manage Linen
  + **3.1.1** The system will add new linen to the table of available linen.
  + **3.1.2** The system will update the details of the linen.
  + **3.1.3** The system will remove linen from the table of available linen.
* **3.2** ManageOrder
  + **3.2.1** The system will log orders from the customer.
  + **3.2.2** The system will cancel an order from the customer.
  + **3.2.3** The system will dispatch the linen for a required order.
  + **3.2.4** The system will collect the laundry the customer wants cleaned.
  + **3.2.5** The system will record the amount owed by the customer and if they have paid.
* **3.3** Manage Customers
  + **3.3.1** The system will add customer details to a table of customers.
  + **3.3.2** The system will update a customer’s information.
  + **3.3.3** The system will remove any obsolete customer.
* **3.4** Perform Admin
  + **3.4.1** The system will calculate the total yearly earnings of the company.
  + **3.4.2** The system will calculate the total earnings per customer.
  + **3.4.3** The system will track the amount of rejects.
  + **3.4.4** The system will track the cost of the rejects in a given time period.

# System Requirements

LinenSYS will manage all linen to be sold by the company. The Add Linen function will add a new piece of linen to the system so that the customer may make orders for that piece of Linen. The Update Linen function will allow the manager to modify details in the Linen File for a particular piece of linen. The Remove Linen function will allow the manager to remove a piece of linen from the system, making it unavailable for orders.

LinenSYS will manage all orders coming through the company. Any new order will be logged into the system using the Log Order function. An order may be cancelled by the manager using the Cancel Order function, assuming the order has not been delivered. The status of an order may be edited using the Dispatch Delivery function. An order to collect Laundry may be logged in the system using the Collect Laundry function. Finally, a record of payment by the customer may be logged in the system using the Record Payment function.

LinenSYS will manage all customers that make orders to the company. The Add Customer function will add a customer to the system so that the customer may make orders for any Linen. The Update Customer function will allow the manager to modify details in the Customer File for any customer. The Remove Customer function will allow the manager to remove a customer from the system, making them unable to make orders.

LinenSYS will also perform various forms of admin. The Calculate Earnings function will calculate the total earnings of the company for the current year and store it in the Earnings File. The Calculate Customer Earnings function will calculate the total earnings that the company has earned from a given customer and store this information in the Customer File. The Track Rejects function will allow the manager to input any rejected linen from an order and save this information in the Rejects file. The Track Reject Cost file will calculate the amount of money the rejected linen from the current year has cost the company.

## System Level Use Case Diagram

LinenSYS

Manager

Customer

## Manage Linen

### **Add Linen**

This function adds new linen to the system.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Linen** | |
| **Use Case Id** | 4.2.1 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function adds new linen to the system. | |
| **Preconditions** | None | |
| **Trigger** | None | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the Add Linen function.  **Step 4:** The manager enters the following details:   * Linen\_Name * Hire\_Price * Cleaning\_Price * Pack\_Size | **Step 2:** The system determines the next Linen\_ID.  **Step 3:** Display the UI  **Step 5:** The system validates the data:   * All fields must not be blank. * The Linen Name must only contain letters. * The Hire and Cleaning Prices must be a positive number * Hire Price must be greater than Cleaning price * The Pack Size must be a positive whole number and a multiple of 5   **Step 6:** The system sets the Status to Active (“A”).  **Step 7:** The system saves the following data to the Linen File:   * Linen\_ID * Linen\_Name * Hire\_Price * Cleaning\_Price * Pack\_Size * Status   **Step 8:** The system displays a confirmation message.  **Step 9:** The system clears the UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **Field Not Entered** |  | **Step 5:** Blank field detected.  **Step 6:** The system displays an error message: “Error, all fields must be entered.”  **Step 7:** Place cursor on first blank field and return to Step 4. |
| **Invalid Linen Name** |  | **Step 5:** Invalid linen name detected (e.g. numbers/symbols).  **Step 6:** The system displays an error message: “Error, invalid linen name, please re-enter.”  **Step 7:** Place cursor on the linen name field and return to Step 4. |
| **Invalid Hire/Cleaning Price** |  | **Step 5:** Invalid hire/cleaning price detected (e.g. not positive numbers with up to two decimal points).  **Step 6:** The system displays an error message: “Error, invalid hire/ cleaning price, please re-enter.”  **Step 7:** Place cursor at the offending field and return to Step 4. |
| **Invalid Pack Size** |  | **Step 5:** Invalid pack size detected (e.g. not positive whole number, and multiple of five).  **Step 6:** The system displays an error message: “Error, invalid pack size, please re-enter.”  **Step 7:** Place cursor at the pack size field and return to Step 4. |
| **Hire Price less than Cleaning Price** |  | **Step 5:** Detected hire price less than cleaning price.  **Step 6:** The system displays an error message: “Error, hire price cannot be less than cleaning price.”  **Step 7:** Place cursor at the hire price field and return to Step 4. |
| **Conclusions** | The linen is added to the system | |
| **Post conditions** | Orders can now be made for the new linen. | |
| **Business Rules** | The pack size must be a multiple of 5.  The hire/cleaning price cannot be a negative number.  The hire price must be greater than the cleaning price. | |
| **Implementation Constraints** | None | |

### **Update Linen**

This function updates the linen currently in the system.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Linen** | |
| **Use Case Id** | 4.2.2 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function updates linen currently in the system. | |
| **Preconditions** | The Linen is already in the system. | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Update Linen function.  **Step 3:** The manager enters the Linen Name (or part of)  **Step 5:** manager selects the linen to be updated  **Step 7:** The manager updates any of the following fields:   * Hire\_Price * Cleaning\_Price * Pack\_Size | **Step 2:** The system displays the UI and prompts the user for the Linen Name.  **Step 4:** The system retrieves summary details from the Linen File for all items with matching name and display on UI  **Step 6:** The system retrieves all details for the selected linen item from the Linen File and displays on the UI for updating  **Step 8:** The system validates the data:   * All fields must not be blank. * The Linen Name must only contain letters. * The Hire and Cleaning Prices must be a positive number * Hire Price must be greater than Cleaning price * The Pack Size must be a positive whole number and a multiple of 5   **Step 9:**  The system saves the new data to the Linen file.  **Step 10:** The system displays a confirmation message.  **Step 11:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Linen Name Not Found** |  | **Step 4:** Linen name not found in the linen file.  **Step 5:** Display an error message: “Error, linen name not found in file, please re-enter.”  **Step 6:** Place cursor on linen name field and return to Step 4. |
| **Invalid Hire/Cleaning Price** |  | **Step 8:** Invalid hire/cleaning price detected (e.g. not positive numbers with up to two decimal points).  **Step 9:** The system displays an error message: “Error, invalid sale/ cleaning price, please re-enter.”  **Step 10:** Place cursor at the offending field and return to Step 8. |
| **Invalid Pack Size** |  | **Step 8:** Invalid pack size detected (e.g. not positive whole number and multiple of 5).  **Step 9:** The system displays an error message: “Error, invalid pack size, please re-enter.”  **Step 10:** Place cursor at the pack size and return to Step 8. |
| **Hire Price less than Cleaning Price** |  | **Step 8:** Detected hire price less than cleaning price.  **Step 9:** The system displays an error message: “Error, hire price cannot be less than cleaning price.”  **Step 10:** Place cursor at the hire price field and return to Step 8. |
| **Conclusions** | The linen is updated in the Linen File | |
| **Post conditions** | None | |
| **Business Rules** | The hire/cleaning price cannot be negative.  The hire price must be greater than the cleaning price.  The pack size must be a multiple of 5. | |
| **Implementation Constraints** | None. | |

### **Remove Linen**

This function removes linen from the system.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Remove Linen** | |
| **Use Case Id** | 4.2.3 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function removes linen currently in the system. | |
| **Preconditions** | The Linen is already in the system. | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Remove Linen function.  **Step 3:** The manager enters the Linen Name (or part of)  **Step 5:** manager selects the linen to be Removed | **Step 2:** The system displays the UI and prompts the user for the Linen Name.  **Step 4:** The system retrieves summary details from the Linen File for all items with matching name and display on UI  **Step 6:** The system sets the status to “I” (Inactive).  **Step 7:**  The system saves the new status of the linen to the Linen File.  **Step 8:** The system displays a confirmation message.  **Step 9:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Linen Name Not Found** |  | **Step 3:** Linen name not found in the linen file.  **Step 4:** Display an error message: “Error, linen name not found in file, please re-enter.”  **Step 5:** Place cursor on linen name field and return to Step 4. |
| **Conclusions** | The linen is removed from the Linen File. | |
| **Post conditions** | The linen can no longer be ordered. | |
| **Business Rules** | The Linen must remain in the Linen file for referential integrity. | |
| **Implementation Constraints** | None. | |

## Manage Orders

### **Log Order**

This function logs an order in the Order file.

Customer

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Log Order** | |
| **Use Case Id** | 4.3.1 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function logs an order in the Order file. | |
| **Preconditions** | The Customer is already in the system.  The Customer has filled in an Order form. | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Log Order function.  **Step 4:** The manager enters the following details:   * Customer\_ID * Linen\_ID(s) * Linen\_Amount * Delivery\_Date | **Step 2:** The system determines the next Order\_ID and sets the Order\_Date to the current date.  **Step 3:** The system displays the UI.  **Step 5:** The system validates the data:   * The Customer\_ID must match a Customer\_ID in the Customer file and must have a status of active (‘A’). * Each Linen\_ID must match a Linen\_ID in the Linen file and must have a status of active (‘A’). * The Linen\_Amount cannot be less than 0 and must be a whole number. * There must be at least one Linen\_ID with a Linen\_Amount greater than 0. * The Delivery\_Date must be at least two working days after the Order\_Date.   **Step 6:** The System sets the Delivery\_Status to “P” (In progress), the Cancel\_Status to “A” (Active), the Payment\_Status to “U” (Unpaid) and the Order\_Type to “D” (Delivery).  **Step 7:**  The system saves the following data to the Order file:   * Order\_ID * Order\_Date * Customer\_ID * Linen\_ID(s) * Linen\_Amount * Delivery\_Date * Delivery\_Status * Cancel\_Status * Order\_Type * Payment\_Status   **Step 8:** The system displays a confirmation message.  **Step 9:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Customer\_ID not matching valid/active customer.** |  | **Step 5:** Invalid/inactive Customer\_ID detected.  **Step 6:** The system displays an error message: “Invalid Customer\_ID detected. Please re-enter”.  **Step 7:** The system places the cursor on the Customer\_ID field and returns to Step 4. |
| **Linen\_ID not matching valid/active customer.** |  | **Step 5:** Invalid/inactive Linen\_ID detected.  **Step 6:** The system displays an error message: “Invalid Linen\_ID detected. Please re-enter”.  **Step 7:** The system places the cursor on the Linen\_ID field and returns to Step 4. |
| **Linen\_Amount less than 0** |  | **Step 5:** Linen\_Amount less than 0 detected.  **Step 6:** The system displays an error message: “Invalid Linen\_Amount detected. Please re-enter”.  **Step 7:** The system places the cursor on the Linen\_Amount field and returns to Step 4. |
| **No Linen\_Amount greater than 0** |  | **Step 5:** No Linen amount greater than 0 detected.  **Step 6:** The system displays an error message: “Error, there must be at least one Linen\_Amount greater than 0”.  **Step 7:** The system places the cursor on the Linen\_Amount field and returns to Step 4. |
| **Invalid Delivery\_Date** |  | **Step 5:** Invalid Delivery\_Date detected (e.g. less than two working days after the Order\_Date).  **Step 6:** The system displays an error message: “Error, the Delivery\_Date must be at least two working days after the Order\_Date”.  **Step 7:** The system places the cursor on the Linen\_Amount field and returns to Step 4. |
| **Conclusions** | The order is Logged in the Order file. | |
| **Post conditions** | None. | |
| **Business Rules** | The order cannot be made by a customer who is not active.  The customer cannot make an order for linen that is not active.  The Delivery\_Date must be at least two working days after the Order\_Date. | |
| **Implementation Constraints** | None. | |

### **Cancel Order**

This function cancels an order in the Order file.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Cancel Order** | |
| **Use Case Id** | 4.3.2. | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function cancels an order in the Order file. | |
| **Preconditions** | The Order is already in the system.  The Delivery status is “P” (In progress). | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Cancel Order function.  **Step 3:** The manager enters the Order\_ID or Order\_Date  **Step 5:** manager selects the order to be cancelled | **Step 2:** The system displays the UI and prompts the user for the Order\_ID or Order\_Date.  **Step 4:** The system retrieves summary details from the Order file for all orders with matching Order\_ID or Order\_Date and display on UI.  **Step 6:** The system sets the Cancel\_Status to “C” (Cancelled).  **Step 7:**  The system saves the new status of the order to the Order File.  **Step 8:** The system displays a confirmation message.  **Step 9:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Order\_ID and Order\_Date not found** |  | **Step 4:** Order\_ID and Order\_Date not found in the Order file.  **Step 5:** Display an error message: “Error, no matching orders, please re-enter.”  **Step 6:** Place cursor on Order\_ID field and return to Step 3. |
| **Conclusions** | The order is cancelled and will not be delivered. | |
| **Post conditions** | None. | |
| **Business Rules** | An order cannot be cancelled if it has already been delivered. | |
| **Implementation Constraints** | None. | |

### **Dispatch Delivery**

This function updates the status of an order to delivered.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Dispatch Delivery** | |
| **Use Case Id** | 4.3.3 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function updates the status of an order to delivered. | |
| **Preconditions** | The delivery is already in the system and ready to be delivered. | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Dispatch Delivery function.  **Step 3:** The manager enters the Order\_ID (or part of)  **Step 5:** The manager selects the delivery to be dispatched. | **Step 2:** The system displays the UI and prompts the user for the Order\_ID.  **Step 4:** The system retrieves summary details from the Order File for all orders with matching ID and display on UI  **Step 6:** The System Retrieves the full details of the order from the Order file.  **Step 7:**  The system changes the Delivery\_Status to “D” (Delivered), the Cancel\_Status to “D” (Delivered) and the Delivery\_Date to the System date.  **Step 8:** The system displays a confirmation message.  **Step 9:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Order\_ID not found** |  | **Step 4:** Order\_ID not found in the Order file.  **Step 5:** Display an error message: “Error, no matching orders, please re-enter.”  **Step 6:** Place cursor on Order\_Name field and return to Step 3. |
| **Conclusions** | The Delivery\_Status of an order has changed, and the order can no longer be cancelled. | |
| **Post conditions** | None | |
| **Business Rules** | The Order must remain in the Order file for referential integrity. | |
| **Implementation Constraints** | None. | |

### **Collect Laundry**

This function logs an order for Laundry Collection in the System.

Customer

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Collect Laundry** | |
| **Use Case Id** | 4.2.3 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function logs an order for Laundry Collection in the System. | |
| **Preconditions** | The Customer has filled in a Laundry form. | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Collect Laundry function.  **Step 4:** The manager enters the following details:   * Customer\_ID * Linen\_ID(s) * Linen\_Amount * Delivery\_Date | **Step 2:** The system determines the next Order\_ID and sets the Order\_Date to the current date.  **Step 3:** The system displays the UI.  **Step 5:** The system validates the data:   * The Customer\_ID must match a Customer\_ID in the Customer file and must have a status of active (‘A’). * Each Linen\_ID must match a Linen\_ID in the Linen file and must have a status of active (‘A’). * The Linen\_Amount cannot be less than 0 and must be a whole number. * There must be at least one Linen\_ID with a Linen\_Amount greater than 0. * The Delivery\_Date must be at least two working days after the Order\_Date.   **Step 6:** The System sets the Delivery\_Status to “P” (In progress), the Cancel\_Status to “A” (Active) and the Order\_Type to “L” (Laundry).  **Step 7:**  The system saves the following data to the Order file:   * Order\_ID * Order\_Date * Customer\_ID * Linen\_ID(s) * Linen\_Amount * Delivery\_Date * Delivery\_Status * Cancel\_Status   **Step 8:** The system displays a confirmation message.  **Step 9:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Customer\_ID not matching valid/active customer.** |  | **Step 5:** Invalid/inactive Customer\_ID detected.  **Step 6:** The system displays an error message: “Invalid Customer\_ID detected. Please re-enter”.  **Step 7:** The system places the cursor on the Customer\_ID field and returns to Step 4. |
| **Linen\_ID not matching valid/active customer.** |  | **Step 5:** Invalid/inactive Linen\_ID detected.  **Step 6:** The system displays an error message: “Invalid Linen\_ID detected. Please re-enter”.  **Step 7:** The system places the cursor on the Linen\_ID field and returns to Step 4. |
| **Linen\_Amount less than 0** |  | **Step 5:** Linen\_Amount less than 0 detected.  **Step 6:** The system displays an error message: “Invalid Linen\_Amount detected. Please re-enter”.  **Step 7:** The system places the cursor on the Linen\_Amount field and returns to Step 4. |
| **No Linen\_Amount greater than 0** |  | **Step 5:** No Linen amount greater than 0 detected.  **Step 6:** The system displays an error message: “Error, there must be at least one Linen\_Amount greater than 0”.  **Step 7:** The system places the cursor on the Linen\_Amount field and returns to Step 4. |
| **Invalid Delivery\_Date** |  | **Step 5:** Invalid Delivery\_Date detected (e.g. less than two working days after the Order\_Date).  **Step 6:** The system displays an error message: “Error, the Delivery\_Date must be at least two working days after the Order\_Date”.  **Step 7:** The system places the cursor on the Linen\_Amount field and returns to Step 4. |
| **Conclusions** | The Order for Laundry is saved in the Order File. | |
| **Post conditions** | None. | |
| **Business Rules** | None. | |
| **Implementation Constraints** | None. | |

### **Record Payment**

This function records a payment for an order in the System.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Record Payment** | |
| **Use Case Id** | 4.2.3 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function records a payment for an order in the System. | |
| **Preconditions** | There is an order with the Payment\_Status of “U” (Unpaid) | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Record Payment function.  **Step 3:** The manager enters the Order\_ID (or part of)  **Step 5:** manager selects the order to be paid | **Step 2:** The system displays the UI and prompts the user for the Order\_ID.  **Step 4:** The system retrieves summary details from the Order File for all orders with matching ID and display on UI.  **Step 6:** The system sets the Payment\_Status to “P” (Paid).  **Step 7:**  The system saves the new status of the order to the Order File.  **Step 8:** The system displays a confirmation message.  **Step 9:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Order\_ID Not Found** |  | **Step 4:** Order\_ID not found in the Order file.  **Step 5:** Display an error message: “Error, Order\_ID not found in file, please re-enter.”  **Step 6:** Place cursor on Order\_ID field and return to Step 3. |
| **Conclusions** | The Payment\_Status of the Order is now “P” (Paid) | |
| **Post conditions** | None | |
| **Business Rules** | The Order must remain in the Order file for referential integrity. | |
| **Implementation Constraints** | None. | |

## Manage Customers

### **Add Customer**

This function adds a new customer to the system.

Customer

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Customer** | |
| **Use Case Id** | 4.4.1 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function adds a new customer to the system. | |
| **Preconditions** | Customer must fill in a Register Customer form. | |
| **Trigger** | None | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the Add Customer function.  **Step 4:** The manager enters the following details:   * Company\_Name * Contact\_Number * Customer\_Name * Email * Street * Town * County * Eircode | **Step 2:** The system determines the next Customer\_ID.  **Step 3:** Display the UI.  **Step 5:** The system validates the data:   * All fields must not be blank. * The Company Name, Customer Name, Town and County must only contain letters. * The Contact Number must be a valid phone number. * The Email Address must be a valid email address. * The Eircode must be a valid Eircode. * The Street, Town and County must all be a valid address. * The Street, Town and County address must match the Eircode address.   **Step 6:** The system sets the Status to Active (“A”).  **Step 7:** The system sets the Balance and Rejects to 0.  **Step 8:** The system saves the following data to the Linen File:   * Customer\_ID * Company\_Name * Contact\_Number * Customer\_Name * Email * Street * Town * County * Eircode * Status * Balance * Rejects   **Step 9:** The system displays a confirmation message.  **Step 10:** The system clears the UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **Field Not Entered** |  | **Step 5:** Blank field detected.  **Step 6:** The system displays an error message: “Error, all fields must be entered.”  **Step 7:** Place cursor on first blank field and return to Step 4. |
| **Invalid Company Name/ Customer Name/ Town/ County** |  | **Step 5:** Invalid Company name/ Customer Name/ Town/ County detected (e.g. numbers/symbols).  **Step 6:** The system displays an error message: “Error, invalid field detected, please re-enter.”  **Step 7:** Place cursor on the offending field and return to Step 4. |
| **Invalid Contact Number** |  | **Step 5:** Invalid contact number detected (e.g. not a valid phone number).  **Step 6:** The system displays an error message: “Error, invalid contact number, please re-enter.”  **Step 7:** Place cursor at the contact number field and return to Step 4. |
| **Invalid Email** |  | **Step 5:** Invalid email detected.  **Step 6:** The system displays an error message: “Error, invalid email, please re-enter.”  **Step 7:** Place cursor at the email and return to Step 4. |
| **Invalid Eircode** |  | **Step 5:** Invalid Eircode detected (e.g. not in the format “x99 xx9x”).  **Step 6:** The system displays an error message: “Error, invalid Eircode, please re-enter.”  **Step 7:** Place the cursor on the Eircode field and return to Step 4. |
| **Invalid Address** |  | **Step 5:** Invalid address detected (e.g. address does not match a real address).  **Step 6:** The system displays an error message: “Error, not an address, please re-enter.”  **Step 7:** Place the cursor on the street field and return to Step 4. |
| **Address Not Matching Eircode** |  | **Step 5:** Eircode not matching address detected.  **Step 6:** The system displays an error message: “Error, Eircode and address do not match, please re-enter.”  **Step 7:** Place the cursor on the street field and return to Step 4. |
| **Conclusions** | The customer is added to the system | |
| **Post conditions** | Orders can now be made for the new linen by the customer. | |
| **Business Rules** | None | |
| **Implementation Constraints** | None | |

### **Update Customer**

This function updates a customer currently in the system.

Manager

Manager

<<extends>>

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Customer** | |
| **Use Case Id** | 4.4.2 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function updates a customer currently in the system. | |
| **Preconditions** | The Customer is already in the system. | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Update Customer function.  **Step 3:** The manager enters the Customer Name (or part of).  **Step 5:** manager selects the linen to be updated.  **Step 7:** The manager updates any of the following fields:   * Company\_Name * Contact\_Number * Customer\_Name * Email * Street * Town * County * Eircode | **Step 2:** The system displays the UI and prompts the user for the Customer Name.  **Step 4:** The system retrieves summary details from the Customer File for all items with matching name and display on UI.  **Step 6:** The system retrieves all details for the selected customer from the Customer File and displays on the UI for updating  **Step 8:** The system validates the data:   * All fields must not be blank. * The Company Name, Customer Name, Town and County must only contain letters. * The Contact Number must be a valid phone number. * The Email Address must be a valid email address. * The Eircode must be a valid Eircode. * The Street, Town and County must all be a valid address. * The Street, Town and County address must match the Eircode address.   **Step 9:**  The system saves the new data to the Customer file.  **Step 10:** The system displays a confirmation message.  **Step 11:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Field Not Entered** |  | **Step 5:** Blank field detected.  **Step 6:** The system displays an error message: “Error, all fields must be entered.”  **Step 7:** Place cursor on first blank field and return to Step 4. |
| **Invalid Company Name/ Customer Name/ Town/ County** |  | **Step 5:** Invalid Company name/ Customer Name/ Town/ County detected (e.g. numbers/symbols).  **Step 6:** The system displays an error message: “Error, invalid field detected, please re-enter.”  **Step 7:** Place cursor on the offending field and return to Step 4. |
| **Invalid Contact Number** |  | **Step 5: Invalid contact number detected (e.g. not a valid phone number).**  **Step 6: The system displays an error message: “Error, invalid contact number, please re-enter.”**  **Step 7: Place cursor at the contact number field and return to Step 4.** |
| **Invalid Email** |  | **Step 5:** Invalid email detected.  **Step 6:** The system displays an error message: “Error, invalid email, please re-enter.”  **Step 7:** Place cursor at the email and return to Step 4. |
| **Invalid Eircode** |  | **Step 5:** Invalid Eircode detected (e.g. not in the format “x99 xx9x”).  **Step 6:** The system displays an error message: “Error, invalid Eircode, please re-enter.”  **Step 7:** Place the cursor on the Eircode field and return to Step 4. |
| **Invalid Address** |  | **Step 5:** Invalid address detected (e.g. address does not match a real address).  **Step 6:** The system displays an error message: “Error, not an address, please re-enter.”  **Step 7:** Place the cursor on the street field and return to Step 4. |
| **Address Not Matching Eircode** |  | **Step 5:** Eircode not matching address detected.    **Step 6:** The system displays an error message: “Error, Eircode and address do not match, please re-enter.”    **Step 7:** Place the cursor on the street field and return to Step 4. |
| **Conclusions** | The customer is updated in the Customer File | |
| **Post conditions** | None | |
| **Business Rules** | None | |
| **Implementation Constraints** | None | |

### **Remove Customer**

This function removes a customer from the system.

Manager

<<extends>>

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Remove Customer** | |
| **Use Case Id** | 4.4.3 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function removes a customer currently in the system. | |
| **Preconditions** | The Customer is already in the system. | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Remove Customer function.  **Step 3:** The manager enters the Customer Name (or part of).  **Step 5:** The manager selects the customer to be Removed. | **Step 2:** The system displays the UI and prompts the user for the Customer Name.  **Step 4:** The system retrieves summary details from the Linen File for all items with matching name and display on UI.  **Step 6:** The system sets the status to “I” (Inactive).  **Step 7:**  The system saves the new status of the customer to the Customer File.  **Step 8:** The system displays a confirmation message.  **Step 9:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Customer Name Not Found** |  | **Step 5:** Linen name not found in the linen file.  **Step 6:** Display an error message: “Error, linen name not found in file, please re-enter.”  **Step 7:** Place cursor on linen name field and return to Step 4. |
| **Conclusions** | The customer is removed from the Customer File. | |
| **Post conditions** | The customer can no longer make an order. | |
| **Business Rules** | The customer must remain in the Customer file for referential integrity. | |
| **Implementation Constraints** | None. | |

## Perform Admin

### **4.5.1 Calculate Earnings**

This function calculates the yearly earnings for the current year.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Calculate Earnings** | |
| **Use Case Id** | 4.4.3 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function calculates the yearly earnings for the current year. | |
| **Preconditions** | None | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Calculate Earnings function. | **Step 2:** The system gathers all the orders from the Order File with the Payment\_Status of “P” (Paid) and the Order\_Date matching the System date’s year and gathers all the Hire\_Price and Cleaning\_Price for each piece of linen from the Linen File.  **Step 3:** The system counts the amount each piece of linen has been hired and multiplies it by the Hire\_Price for that piece of linen and saves this as the Hired\_Earnings.  **Step 4:** The system counts the amount each piece of linen has been collected for cleaning and multiplies it by the Cleaning\_Price for that piece of linen and saves this as the Laundry\_Earnings.  **Step 5:** The system adds the Hired\_Earnings and the Laundry\_Earnings and saves this as the Total\_Earnings.  **Step 6:** The system displays the Hired\_Earnings, Laundry\_Earnings and Total\_Earnings on the UI.  **Step 7:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The manager can compare the yearly earnings to a previous year’s earnings. | |
| **Post conditions** | None. | |
| **Business Rules** | None. | |
| **Implementation Constraints** | None. | |

### **4.5.2 Calculate Customer Earnings**

This function calculates the total earnings from each customer.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Calculate Customer Earnings** | |
| **Use Case Id** | 4.4.3 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function calculates the total earnings from each customer. | |
| **Preconditions** | Each Customer is already in the system. | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Calculate Customer Earnings function. | **Step 2:** The system gathers all the orders from the Order File with the Payment\_Status of “P” (Paid) and gathers all the Hire Price and Cleaning Price for each piece of linen from the Linen File.  **Step 3:** The system counts the amount each piece of linen has been hired by a customer and multiplies it by the Hire Price for that piece of linen and saves this as the [Customer\_ID] \_Hired.  **Step 4:** The system counts the amount each piece of linen has been collected for cleaning and multiplies it by the Cleaning Price for that piece of linen and saves this as the [Customer\_ID] \_Laundry.  **Step 5:** The system adds the [Customer\_ID] \_Hired and the [Customer\_ID] \_Laundry and saves this as the [Customer\_ID] \_Total.  **Step 6:** The system displays the [Customer\_ID] \_Hired, [Customer\_ID] \_Laundry and [Customer\_ID] \_Total for each Customer\_ID on the UI.  **Step 7:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The manager can now see which customer is earning the company the most money. | |
| **Post conditions** | None | |
| **Business Rules** | None. | |
| **Implementation Constraints** | None. | |

### **Track Rejects**

This function inputs rejects from an order into the Rejects File.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Track Rejects** | |
| **Use Case Id** | 4.4.3 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function inputs rejects from an order into the Rejects File. | |
| **Preconditions** | The Linen\_ID and Order\_ID are both in the system | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Track Rejects function.  **Step 3:** The manager enters the Order\_ID (or part of).  **Step 5:** The manager selects the Order to enter the rejects.  **Step 7:** The manager enters the Linen\_ID(s) and Linen\_Amount. | **Step 2:** The system displays the UI and prompts the user for the Order\_ID.  **Step 4:** The system retrieves summary details from the Order File for all items with matching ID and display on UI.  **Step 6:** The System prompts the user for the rejects.  **Step 8:**  The system sets the Reject\_Date to the current system date.  **Step 9:** The System saves the Linen\_ID(s), Linen\_Amount and Reject\_Date in the Rejects file  **Step 10:** The system displays a confirmation message.  **Step 11:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Order\_ID Not Found** |  | **Step 4:** Order\_ID not found in the Order file.  **Step 5:** Display an error message: “Error, Order\_ID not found in file, please re-enter.”  **Step 6:** The system places the cursor on Order\_ID field and return to Step 3. |
| **Linen\_ID Not Found** |  | **Step 8:** Linen\_ID not found in the Linen file.  **Step 9:** Display an error message: “Error, Linen\_ID not found in file, please re-enter.”  **Step 10:** The system places the cursor on Linen\_ID field and return to Step 7. |
| **No Linen\_Amount Greater than 0** |  | **Step 8:** No Linen\_Amount greater than 0 detected.  **Step 9:** The system displays an error message: “Error, there must be at least one Linen\_Amount greater than 0”.  **Step 10** The system places the cursor on the Linen\_Amount field and returns to Step 7. |
| **Linen\_Amount Less than 0** |  | **Step 8:** Linen\_Amount less than 0 detected.  **Step 9:** The system displays an error message: “Invalid Linen\_Amount detected. Please re-enter”.  **Step 10:** The system places the cursor on the Linen\_Amount field and returns to Step 7. |
| **Linen\_Amount in Rejects greater than Linen\_Amount in Order** |  | **Step 8:** Linen\_Amount in rejects greater than Linen\_Amount in order detected.  **Step 9:** The system displays an error message: “Error, rejects must be less than amount ordered. Please re-enter”.  **Step 10:** The system places the cursor on the Linen\_Amount field and returns to Step 7. |
| **Conclusions** | The Reject File is now updated. | |
| **Post conditions** | None. | |
| **Business Rules** | The Linen\_Amount must be greater than 0 and less than the Linen\_Amount in the order. | |
| **Implementation Constraints** | None. | |

### **Track Reject Cost**

This function tracks the yearly cost of rejects per piece of Linen for the current year.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Track Reject Cost** | |
| **Use Case Id** | 4.4.3 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function tracks the yearly cost of rejects per piece of Linen for the current year. | |
| **Preconditions** | None. | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Track Reject Cost function. | **Step 2:** The system retrieves the Linen\_ID(s) and Linen\_Amount for each Reject matching the current system date’s year from the Rejects file.  **Step 3:** The system retrieves the Hire\_Price and Cleaning\_Price for each piece of linen from the Linen file  **Step 4:** The system counts the Linen\_Amount and multiplies it by the Hire\_Price for that piece of linen and saves this as the Hired\_Lost.  **Step 4:** The system counts the Linen\_Amount and multiplies it by the Cleaning\_Price for that piece of linen and saves this as the Laundry\_Lost.  **Step 5:** The system gets the average of the Hired\_Lost and the Laundry\_Lost and saves this as the Average\_Lost.  **Step 6:** The system adds the Hired\_Lost and the Laundry\_Lost and saves this as the Max\_Lost.  **Step 6:** The system displays the Hired\_Lost, Laundry\_Lost, Average Lost and Max\_Lost on the UI.  **Step 7:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The manager can now see how much money was lost to rejects for the current year. | |
| **Post conditions** | None. | |
| **Business Rules** | None. | |
| **Implementation Constraints** | None. | |

# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

## Level-1 DFD

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

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

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

# Data Model (Class Diagram)

Brief introduction……

## Class Diagram

Object Model – UML Class Diagram

Class diagram shows objects & attributes

## Relational Schema

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

## Database Schema

A definition of the database to be implemented.

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

# Conclusion

# Appendices

## Appendix A – Title

## Appendix B – Title

Might include:

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