Requirements Engineering

Linen Order SYS

Submitted By: Michael Edgar – T00194492

Computing with Games Development

Date Submitted: 12/12/2017

**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

In this project, I aim to create a linen ordering system for a linen company. In this system, I aim to allow the user to work on four main components: Linen, Customers, Orders and Admin. The user will be able to add, update and remove linen. The user will be able to perform the same functions with customers, and use the data from the customers and linen to create orders for linen and laundry. The user will be able to cancel and dispatch the orders. Finally, the user will be able to calculate total earnings; along with earnings per customer and record rejects, while tracking their cost on the company.

# 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.
* **3.3** 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 record 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.

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 Record 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 3:** The manager enters the following details:   * Linen\_Name * Linen\_Code * Hire\_Price * Cleaning\_Price * Reject\_Price * Pack\_Size | **Step 2:** Display the UI  **Step 4:** The system validates the data:   * All fields must not be blank. * The Linen Name must only contain letters. * The Linen Code must be unique and only contain letters. * The Hire, Cleaning and Reject Prices must be a positive number * Hire Price must be greater than Cleaning price and Cleaning Price must be greater than Reject Price. * The Pack Size must be a positive whole number and a multiple of 5   **Step 5:** The system sets the Linen\_Status to Active (“A”).  **Step 6:** The system saves the following data to the Linen File:   * Linen\_Code * Linen\_Name * Hire\_Price * Cleaning\_Price * Reject\_Price * Pack\_Size * Linen\_Status   **Step 7:** The system displays a confirmation message.  **Step8:** The system clears the UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **Field Not Entered** |  | **Step 4:** Blank field detected.  **Step 5:** The system displays an error message: “Error, all fields must be entered.”  **Step 6:** Place cursor on first blank field and return to Step 3. |
| **Invalid Linen Code** |  | **Step 4:** Invalid linen code detected (e.g. not unique, contains numbers or symbols)  **Step 5:** The system displays an error message: “Error, invalid linen code, please re-enter”  **Step 6:** Place cursor on linen code field and return to Step 3. |
| **Invalid Linen Name** |  | **Step 4:** Invalid linen name detected (e.g. numbers/symbols).  **Step 5:** The system displays an error message: “Error, invalid linen name, please re-enter.”  **Step 6:** Place cursor on the linen name field and return to Step 3. |
| **Invalid Hire/Cleaning/ Reject Price** |  | **Step 4:** Invalid hire/cleaning/reject price detected (e.g. not positive numbers with up to two decimal points).  **Step 5:** The system displays an error message: “Error, invalid hire/cleaning/reject price, please re-enter.”  **Step 6:** Place cursor at the offending field and return to Step 3. |
| **Invalid Pack Size** |  | **Step 4:** Invalid pack size detected (e.g. not positive whole number, and multiple of five).  **Step 5:** The system displays an error message: “Error, invalid pack size, please re-enter.”  **Step 6:** Place cursor at the pack size field and return to Step 3. |
| **Hire Price less than Cleaning Price/ Cleaning Price less than Reject Price** |  | **Step 4:** Detected hire price less than cleaning price/cleaning price less than reject price.  **Step 5:** The system displays an error message: “Error, hire price cannot be less than cleaning price.” Or “Error, hire price cannot be less than cleaning price.”  **Step 6:** Place cursor at the hire/cleaning price field and return to Step 3. |
| **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/reject price cannot be a negative number.  The hire price must be greater than the cleaning price.  The cleaning price must be greater than the reject 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 Code.  **Step 5:** manager selects the linen to be updated.  **Step 7:** The manager updates any of the following fields:   * Hire\_Price * Cleaning\_Price * Reject\_Price * Pack\_Size * Linen\_Name | **Step 2:** The system displays the UI and prompts the user for the Linen Code.  **Step 4:** The system retrieves summary details from the Linen File for all items with matching code 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, Cleaning and Reject Prices must be a positive number * Hire Price must be greater than Cleaning price and Cleaning Price must be greater than Reject 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 Code Not Found** |  | **Step 4:** Linen code not found in the linen file.  **Step 5:** Display an error message: “Error, linen code not found in file, please re-enter.”  **Step 6:** Place cursor on linen code field and return to Step 4. |
| **Invalid Hire/Cleaning/Reject Price** |  | **Step 8:** Invalid hire/cleaning/reject price detected (e.g. not positive numbers with up to two decimal points).  **Step 9:** The system displays an error message: “Error, invalid hire/cleaning/reject 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/Cleaning Price less than Reject Price** |  | **Step 8:** Detected hire price less than cleaning price/ cleaning price less than reject price.  **Step 9:** The system displays an error message: “Error, hire price cannot be less than cleaning price.” Or “Error, cleaning price cannot be less than reject 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 pack size must be a multiple of 5.  The hire/cleaning/reject price cannot be a negative number.  The hire price must be greater than the cleaning price.  The cleaning price must be greater than the reject price. | |
| **Implementation Constraints** | None. | |

### **Remove Linen**

This function removes linen from the system.

Manager

|  |  |  |
| --- | --- | --- |
| **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 Code.  **Step 5:** manager selects the linen to be Removed. | **Step 2:** The system displays the UI and prompts the user for the Linen Code.  **Step 4:** The system retrieves summary details from the Linen File for all items with matching code and display on UI.  **Step 6:** The system sets the Linen\_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 Code Not Found** |  | **Step 3:** Linen code not found in the linen file.  **Step 4:** Display an error message: “Error, linen code not found in file, please re-enter.”  **Step 5:** Place cursor on linen code 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 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 Customer\_Status to Active (“A”).  **Step 7:** The system sets the 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 * Customer\_Status * 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

|  |  |  |
| --- | --- | --- |
| **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 Customer\_Status to “I” (Inactive).  **Step 7:**  The system saves the new Customer\_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. | |

## 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 Customer Name (or part of).  **Step 6:** manager selects the customer required for the order.  **Step 9:** The manager enters the following details:   * Linen Code * Linen\_Amount * Delivery\_Date | **Step 2:** The system determines the next Order\_ID, determines the next Order\_Item and sets the Order\_Date to the current date.  **Step 3:** The system displays the UI.  **Step 5:** The system retrieves summary details from the Customer File for all customers with matching name and display on UI.  **Step 7:** The system retrieves all customer details from the customer file and displays on UI  **Step 8:** The system retrieves all linen details from the linen file and displays on UI.  **Step 10:** The system validates the data:   * The Customer Name must match a Customer Name in the Customer file and must have a status of active (‘A’). * Each Linen\_Code must match a Linen\_Code 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\_Code with a Linen\_Amount greater than 0. * The Delivery\_Date must be at least two working days after the Order\_Date.   **Go to step 8**  **Step 11:** The System sets the order Status to “P” (In progress) and the Order\_Type to “D” (Delivery).  **Step 12:**  The system saves the following data to the Order file:   * Order\_ID * Order\_Date * Customer\_ID * Delivery\_Date * Status * Order\_Type   **Step 13: For each item ordered** The System saves the following data to the Order Items File:   * Order\_Item * Linen\_Code(s) * Linen\_Amount * Order\_ID   **Step 14:** The system displays a confirmation message.  **Step 15:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Customer\_Name not matching valid/active customer.** |  | **Step 5:** Invalid/inactive Customer\_Name detected.  **Step 6:** The system displays an error message: “Invalid Customer\_Name detected. Please re-enter”.  **Step 7:** The system places the cursor on the Customer\_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 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 (or part of).  **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 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 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\_ID field and return to Step 3. |
| **Conclusions** | The 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 Customer Name (or part of).  **Step 6:** manager selects the customer required for the order.  **Step 9:** The manager enters the following details:   * Linen\_Amount * Delivery\_Date | **Step 2:** The system determines the next Order\_ID, determines the next Order\_Item and sets the Order\_Date to the current date.  **Step 3:** The system displays the UI.  **Step 5:** The system retrieves summary details from the Customer File for all customers with matching name and display on UI.  **Step 7:** The System Retrieves the full details of the customer from the Customer file and displays on UI for viewing.  **Step 8:** The system retrieves details of displays each Linen\_Name from Linen file and displays on UI.  **Step 11:** The system validates the data:   * The Customer Name must match a Customer Name in the Customer file and must have a status of active (‘A’). * Each Linen\_Code must match a Linen\_Code 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\_Code with a Linen\_Amount greater than 0. * The Delivery\_Date must be at least two working days after the Order\_Date.   **Step 12:** The System sets the Status to “P” (In progress) and the Order\_Type to “L” (Laundry).  **Step 13:**  The system saves the following data to the Order file:   * Order\_ID * Order\_Date * Customer\_ID * Delivery\_Date * Status * Order\_Type   **Step 14:** The System saves the following data to the Order Items File:   * Order\_Item * Linen\_Code (s) * Linen\_Amount * Order\_ID   **Step 15:** The system displays a confirmation message.  **Step 16:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Customer\_Name not matching valid/active customer.** |  | **Step 5:** Invalid/inactive Customer\_Name detected.  **Step 6:** The system displays an error message: “Invalid Customer\_Name detected. Please re-enter”.  **Step 7:** The system places the cursor on the Customer\_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. | |

## 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 order IDs and total Prices from the Order File with the Order\_Date matching the System date’s year, gathers all the prices from the Order Items file with Order IDs matching the Order IDs in the Order File 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 Order\_Date matching the System date’s year, gathers all the Orders from the Order Items file with Order IDs matching the Order IDs in the Order File 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. | |

### **Record Rejects**

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

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Record 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\_Code and Order\_ID are both in the system | |
| **Trigger** | No | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Record 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 8:** The manager enters the Linen\_Code(s) and Reject\_Qty. | **Step 2:** The system determines the next Reject\_ID 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 Retrieves the full details of the order from the Order file.  **Step 7:** The System prompts the user for the rejects.  **Step 9:**  The system sets the Reject\_Date to the current system date.  **Step 10:** The system calculates the reject attribute by multiplying the Reject\_Price by the Reject\_Qty for each rejected Linen\_Code and adding it to the current value of the reject attribute.  **Step 11:** The System saves the Reject\_ID, Linen\_Code(s), Reject\_Qty, Reject\_Price, Order\_ID and Reject\_Date in the Rejects file.  **Step 12:** The System saves the new reject attribute in the Customer file.  **Step 13:** The system displays a confirmation message.  **Step 14:** 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\_Code Not Found** |  | **Step 9:** Linen\_Code not found in the Linen file.  **Step 10:** Display an error message: “Error, Linen Code not found in file, please re-enter.”  **Step 11:** The system places the cursor on Linen\_Code field and return to Step 7. |
| **No Linen\_Amount Greater than 0** |  | **Step 9:** No Linen\_Amount greater than 0 detected.  **Step 10:** The system displays an error message: “Error, there must be at least one Linen\_Amount greater than 0”.  **Step 11:** The system places the cursor on the Linen\_Amount field and returns to Step 7. |
| **Linen\_Amount Less than 0** |  | **Step 9:** Linen\_Amount less than 0 detected.  **Step 10:** The system displays an error message: “Invalid Linen\_Amount detected. Please re-enter”.  **Step 11:** 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 9:** Linen\_Amount in rejects greater than Linen\_Amount in order detected.  **Step 10:** The system displays an error message: “Error, rejects must be less than amount ordered. Please re-enter”.  **Step 11:** 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\_Code(s) and Linen\_Amount for each Reject matching the current system date’s year from the Rejects file and the Reject\_Price for each Linen\_Code in the Linen File.  **Step 3:** The system counts the Linen\_Amount and multiplies it by the Reject\_Price for that piece of linen and saves this as the Reject\_Lost.  **Step 5:** The system gets the average of the Reject\_Lost for each piece of Linen and saves this as the Average\_Lost.  **Step 6:** The system counts the Linen\_Amount of each piece of Linen and saves the greatest Linen\_Amount as the Most\_Lost.  **Step 6:** The system displays the Reject\_Lost, Average Lost and Most\_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:

## DFD Elements

External Entities

Customer

Data Stores

D1 Linen File

D2 Customer File

D3 Order File

D4 Order Items File

D5 Reject File

Processes

P1 Manage Linen P2 Manage Customers

P1.1 Add Linen P2.1 Add Customer

P1.2 Update Linen P2.2 Update Customer

P1.3 Remove Linen P2.3 Remove Customer

P3 Manage Orders P4 Perform Admin

P3.1 Log Order P4.1 Calculate Earnings

P3.2 Cancel Order P4.2 Calculate Customer Earnings

P3.3 Dispatch Delivery P4.3 Record Rejects

P3.4 Collect Laundry P4.4 Track Reject Cost

## Level-0 DFD

LinenSys

Make Orders/ Register details

Customer

Dispatch Deliveries

## Level-1 DFD

Customer Details

P2

Manage Customers

D2

Customer File

Customer

Customer Details

P1

Manage Linen

Linen Details

D1

Linen File

Linen Details

Linen Names

Customer Details

Customer Details

D2

Customer File

P3

Manage Orders

Order Details

P4

Perform Admin

D4

Order Items File

Order Details

Order Details

D3

Order File

Order Details

Reject Details

D5

Reject File

## Level-2 DFD Manage Linen

Linen Details

P1

Add Linen

D1

Linen File

Linen Details

P3

Remove Linen

Linen Details

Linen Details

Linen Details

P2

Update Linen

## Level-2 DFD Manage Customers

P3

Remove Customer

D2

Customer File

Customer Details

Customer Details

Customer Details

P1

Add Customer

Customer Details

Customer Details

Customer Details

P2

Update Customer

Customer Details

Customer Details

Customer

## Level-2 DFD Manage Orders

Customer Details

P1

Log Order

Customer Details

P4

Collect Laundry

D2

Customer File

Linen Details

Linen Details

D1

Linen File

Order Details

Order Details

Order Details

D3

Order File

Order Details

D4

Order Items File

Order Details

Order Details

Customer

Order Details

Order Details

P2

Cancel Order

D3

Order File

Order Details

D4

Order Items File

P3

Dispatch Delivery

Order Details

Order Details

## Level-2 DFD Perform Admin

P1

Calculate Earnings

P3

Track Rejects

Order Details

Linen Details

D3

Order File

Linen Details

Order Details

Customer Details

D3

Order File

Order Details

Reject Details

Order Details

D1

Linen File

Linen Details

D4

Order Items File

P2

Calculate Customer Earnings

D2

Customer File

D5

Reject File

Customer Details

Order Details

Reject Details

P4

Track Reject Cost

D1

Linen File

Reject Prices

# Data Model (Class Diagram)

Brief introduction……

## 6.1. Class Diagram

## Relational Schema

Linen (Linen\_Code, Linen\_Name, Hire\_Price, Cleaning\_Price, Reject\_Price, Pack\_Size, Linen\_Status)

Customers (Customer\_ID, Company\_Name, Contact\_Number, Customer\_Name, Email, Street, Town, County, Eircode, Customer\_Status, Rejects)

Orders (Order\_ID, Order\_Date, Delivery\_Date, Status, Order\_Type, Customer\_ID)

Order Items (Order\_Item, Linen\_Amount, Linen\_Code, Order\_ID)

Rejects (Reject\_ID, Reject\_Date, Reject\_Qty, Linen\_Code, Order\_ID)

## Database Schema

**Schema:** **Linen Sys**

**Relation: Linen**

Attributes:

Linen\_Code Char (3) NOT NULL UNIQUE

Linen\_Name Char (30)

Hire\_Price Numeric (6, 2)

Cleaning\_Price Numeric (6, 2)

Reject\_Price Numeric (6, 2)

Pack\_Size Numeric (3)

Linen\_Status Char (1)

**Primary Key:** Linen\_Code

**Relation: Customer**

Attributes:

Customer\_ID Numeric (6) NOT NULL UNIQUE

Company\_Name Char (20)

Contact\_Number Char (10)

Customer\_Name Char (20)

Email Char (30)

Street Char (20)

Town Char (20)

County Char (10)

Eircode Char (8)

Customer\_Status Char (1)

Rejects Numeric (6, 2)

**Primary Key:** Customer\_ID

**Relation: Order**

Attributes:

Order\_ID Numeric (6) NOT NULL UNIQUE

Order\_Date Date

Delivery\_Date Date

Status Char (1)

Order\_Type Char (1)

Customer\_ID Numeric (6)

**Primary Key:** Order\_ID

**Foreign Key:** Customer\_ID References Customer

**Relation: Order Item**

Attributes:

Order\_Item Numeric (6) NOT NULL UNIQUE

Linen\_Amount Numeric (6, 2)

Linen\_Code Char (3)

Order\_ID Numeric (6)

**Primary Key:** Order\_Item

**Foreign Key:** Linen\_Code References Linen

**Foreign Key:** Order\_ID References Order

**Relation: Reject**

Attributes:

Reject\_ID Numeric (6) NOT NULL UNIQUE

Reject\_Date Date

Reject\_Qty Numeric (3)

Linen\_Code Char (3)

Order\_ID Numeric (6)

**Primary Key:** Reject\_ID

**Foreign Key:** Linen\_Code References Linen

**Foreign Key:** Order\_ID References Order

# Conclusion

In this project I successfully created a linen ordering system, with the full functionality I originally aimed to have in it. In future versions however, I would add functions to calculate earnings on a monthly basis and save the earnings to be viewed later. I would also add functionality to see how many orders have been made both in a particular month and for a particular piece of Linen.

# Appendices

## Appendix A – Title

## Appendix B – Title