Requirements Engineering

Restaurant Till System

**Submitted By:** Billy Elsbury (T00224562)

Computing with Games Development

Date Submitted: 09/12/2022

**Table of Contents**

[1. Introduction/overview 4](#_Toc121446766)

[2. Functional Components 5](#_Toc121446767)

[3. User Requirements 5](#_Toc121446768)

[3.1. RestaurantSYS will process Menu Items 5](#_Toc121446769)

[3.2. RestaurantSYS will manage Orders 5](#_Toc121446770)

[3.3. RestaurantSYS will perform Administrative Reporting 5](#_Toc121446771)

[4. System Requirements 6](#_Toc121446772)

[4.1. System Level Use Case Diagram 6](#_Toc121446773)

[4.2. Manage Menu 7](#_Toc121446774)

[**4.2.1.** **Add Item** 7](#_Toc121446775)

[**4.2.2.** **Edit Item** 9](#_Toc121446776)

[**4.2.3.** **Remove Item** 11](#_Toc121446777)

[**4.2.4.** **Query Item** 12](#_Toc121446784)

[4.3. Manage Orders 14](#_Toc121446785)

[**4.3.1.** **Place Order** 14](#_Toc121446786)

[**4.3.2.** **Edit Order** 15](#_Toc121446787)

[**4.3.3.** **Cancel Order** 16](#_Toc121446788)

[**4.3.4.** **Pay Bill** 17](#_Toc121446789)

[4.4. Perform Admin Functions 18](#_Toc121446790)

[**4.4.1.** **Item Analysis** 18](#_Toc121446791)

[**4.4.2.** **Revenue Analysis** 19](#_Toc121446792)

[5. System Model 21](#_Toc121446793)

[5.1. Level-0 DFD 21](#_Toc121446794)

[5.2. Level-1 DFD 21](#_Toc121446795)

[5.3. Level-2 DFD (Process P1: Menu Items) 22](#_Toc121446796)

[5.4. Level-2 DFD (Process P2: Orders) 22](#_Toc121446797)

[5.5. Level-2 DFD (Process P3: Admin) 23](#_Toc121446798)

[6. Data Model (Class Diagram) 23](#_Toc121446799)

[6.1. Class Diagram 23](#_Toc121446800)

[6.2. Relational Schema 24](#_Toc121446801)

[6.3. Database Schema 24](#_Toc121446802)

[7. Conclusion 25](#_Toc121446803)

[8. Appendices 26](#_Toc121446804)

[8.1. Appendix A – Menu Item Analysis 26](#_Toc121446805)

[8.2. Appendix B – Yearly Revenue Analysis 26](#_Toc121446806)

[8.1. Appendix C – Declaration of Originality 27](#_Toc121446807)

# Introduction/overview

This documentation will outline the operations and functions of a Restaurant ordering System. The system includes three distinct Components to facilitate the necessary processes.

The System will allow the Restaurant Management to add, amend and delete food/drink items in the System. Staff will be able to view the existing menu items. Staff will also be able to create, edit, cancel, and close orders. If a theoretical order was taken by a staff member, this order could then be added into the System and would be comprised of items from the System’s menu items.

The Restaurant manager will also be able to generate sales statistics such as an analysis of items sold to display what menu items are selling the most. The manager could also invoke a Revenue analysis function that would depict revenue achieved per month for a selected year.

The Restaurant till System will require three separate data stores, a MenuItems file, an Orders file, and an OrderMenuItems file.

* The MenuItems file contains item information such as an Item’s ID, whether it is a food item, drink, or dessert, brief description, name, and unity price.
* The Orders file contains order information such as an order’s ID, the date the order was entered into the System, the total price of the order, and the status to represent whether an order is still open or has been closed and is unavailable.
* Finally, the OrderMenuItems file prevents breaking cardinality rules in database design, this is necessary as one Order could contain many of the same menu items. Similarly, a singular menu item may be apart of many different orders. This would cause a M:M/many to many relationship. The OrderMenuItems file can act as a junction between the 2 primary files to avoid this.

# Functional Components

# User Requirements

## RestaurantSYS will process Menu Items

* + 1. RestaurantSYS will add a menu item
    2. RestaurantSYS will edit a menu item
    3. RestaurantSYS will remove a menu item

## RestaurantSYS will manage Orders

* + 1. RestaurantSYS will place an order
    2. RestaurantSYS will edit an order
    3. RestaurantSYS will cancel an order
    4. RestaurantSYS will process payment of a bill

## RestaurantSYS will perform Administrative Reporting

* + 1. RestaurantSYS will produce an Item sales analysis report
    2. RestaurantSYS will produce a Revenue analysis report

# System Requirements

The top-level modules of this system include Menu Items, Orders, and Admin.

Menu Items are the food, beverage and dessert Items contained within the Restaurant System’s menu, the Manager will be able to add new items to the menu, edit existing menu items, and remove items from the menu.

Orders outline the Orders saved on the till System. Restaurant staff can place new orders in the system, edit existing orders, cancel/void existing orders as well as pay the bill and subsequently close an open order.

Admin processes include Item Analysis which outlines a menu item’s statistics such as sale frequency. The Revenue Analysis module can provide a Revenue analysis report of the System’s sales and profits over a period.

## System Level Use Case Diagram

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

Restaurant System

Manage Orders

Manage Menu

Manager

Waiter

Customer

Waiter

Manager

Perform Admin

## Manage Menu

This component allows the manager to add a menu item, edit an existing menu item, and delete an existing menu item.

### **Add Item**

This function adds a new food or drink item into the menu system.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add Item | |
| **Use Case Id** |  | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Restaurant Staff, Chef | |
| **Description** | This function allows the Manager to add new menu items into the system. | |
| **Preconditions** | The menu item cannot already exist | |
| **Trigger** |  | |
|  |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Add Item function  **Step 4:** Manager enters new menu item details:   * Type (Char)   [F/B/D] For food, beverage, or dessert   * Name (varChar) * Description (varChar) * Price (Number) | **Step 2:** System retrieves item types from the ‘**Types File’** and loads on UI  **Step 3:** System displays UI  **Step 5:** System performs validations   * No item details are null * Name must not be numeric * Description must not be numeric * Price must be numeric, > 0   **Step 6:** System assigns the latest item a unique MenuItemId  **Step 5:** System defaults the item’s status to available (‘A’)  **Step 7:** Save data in Menu Items File:   * MenuItemID (Number) * Type (Char) * Name (varChar) * Description (varChar) * Price (Number) * Status (‘A’)   **Step 8:** Display confirmation message  **Step 7: System** UI is reset |
| **Alternate Scenarios** | **Actor** | **System** |
|  | **Step 4:** Invalid data entered | **Step 5:** System validation fails  **Step 6:** Prompt Manager with error message  **Step 7:** UI is reset to **step 4.** |
| **Conclusions** | A new Menu Item has been added to the System’s menu. | |
| **Post conditions** | The new Menu Item can be ordered. | |
| **Business Rules** | Duplicates of a menu Item cannot be entered. | |
| **Implementation Constraints** |  | |

### **Edit Item**

This function allows the Manager to edit an existing item in the till system’s menu.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Edit Item | |
| **Use Case Id** |  | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Restaurant Staff, Chef | |
| **Description** | This function allows the Manager to edit existing menu items from the till system. | |
| **Preconditions** | The menu item must exist before it can be edited | |
| **Trigger** |  | |
|  |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Edit Item function  **Step 4:** Manager selects the MenuItemID of the item which they wish to edit.  **Step 6:** Manager alters existing menu item details:   * Type (Char)   [F/B/D] For food, beverage, or dessert   * Name (varChar) * Description (varChar) * Price (Number) | **Step2:** System retrieves summary of available items ‘Menu Items File’  **Step 3:** System displays UI and awaits further input  **Step 5:** System retrieves full details of selected item from ‘Menu Items File’ and  **Step 7:** System performs validation on the edited menu item.   * No item details are null * Name must not be numeric * Description must not be numeric * Price must be numeric, > 0   **Step 8:** System saves updates in the ‘Menu Items File’  **Step 9:** Display confirmation message  **Step 10:** SystemUI is reset |
| **Alternate Scenarios** | **Manager** | **System** |
|  | **Step 4:** Invalid data entered | **Step 5:** System validation fails  **Step 6:** Prompt Manager with error message  **Step 7:** UI is reset to step 5. |
| **Conclusions** | A Menu Item has been edited and updated in the System’s menu. | |
| **Post conditions** | The altered Menu Item can be selected by staff. | |
| **Business Rules** |  | |
| **Implementation Constraints** | A menu item could be edited while it is being referenced by an order, this could cause logical errors. IE: An item’s price is changed, does the value of an order change with it | |

### **Remove Item**

This function allows the Manager to Remove an existing item from the till system’s menu.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Remove Item | |
| **Use Case Id** |  | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Restaurant Staff, Chef | |
| **Description** | This function allows the Manager to remove existing menu items from the till system. | |
| **Preconditions** | The menu item must exist before it can be deleted | |
| **Trigger** |  | |
|  |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Remove Item function  **Step 4:** Manager selects from the which menu Items to remove from the list of menu items. | **Step2:** System retrieves item details from **‘Menu Items File’**  **Step 3:** System displays UI and awaits further input  **Step 5:** System sets the item’s status to unavailable (‘U’) in the **‘Menu Items File’**  **Step 8:** Display confirmation message  **Step 6:** SystemUI is reset |
| **Alternate Scenarios** | **Manager** | **System** |
|  |  |  |
| **Conclusions** | The menu Item is unavailable in the System’s menu. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** | A menu item could be removed while it is being referenced by an order, this could cause logical errors. IE: An item is removed, the System must handle an order that contained that menu item correctly | |



### **Query Item**

This function allows the Manager to Query and view an existing item from the till system’s menu.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Query Item | |
| **Use Case Id** |  | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the Manager to query an existing menu item from the till system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
|  |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Query Item function  **Step 4:** Manager selects filters to filter the menu items   * unavailable (‘U’)/ available (‘A’) * MenuItemID * Type (Char)   [F/B/D] For food, beverage, or dessert   * Name (varChar) * Description (varChar) * Price (Number) | **Step 2:** System retrieves item details from **‘Menu Items File’**  **Step 3:** System displays UI and awaits further input  **Step 5:** System displays filtered menu items  **Step 6:** SystemUI is reset |
| **Alternate Scenarios** | **Manager** | **System** |
|  |  |  |
| **Conclusions** |  | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Manage Orders

### **Place Order**

This function allows a staff member to place an order into the till System.

Staff

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Place Order | |
| **Use Case Id** |  | |
| **Priority** | High | |
| **Source** | Staff | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** | Chef | |
| **Description** | This function allows the staff to place a new order into the till System | |
| **Preconditions** |  | |
| **Trigger** |  | |
|  |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Staff member invokes the Place Order Function  **Step 4:** Staff member selects which menu Items they would like to add to the order | **Step2:** System retrieves item details from **‘Menu Items File’**  **Step 3:** System displays UI and awaits further input  **Step 5:** System adds timestamp to order   * Order-creation = System date   **Step 5:** System adds new order to the **‘Orders File’**  **Step 5:** System sets the order’s status to open (‘O’) in the **‘orders file’**  **Step 6:** System sends Food Type Items from the order to the kitchen  **Step 8:** Display confirmation message  **Step 6:** SystemUI is reset |
| **Alternate Scenarios** | **Manager** | **System** |
|  |  |  |
| **Conclusions** | A new order has been added to the System. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Edit Order**

This function allows a staff member to edit an existing order in the till System.

Staff

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Place Order | |
| **Use Case Id** |  | |
| **Priority** | High | |
| **Source** | Staff | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** | Chef | |
| **Description** | This function allows the staff to edit an existing order. | |
| **Preconditions** | The order the staff member wishes to edit exists. | |
| **Trigger** |  | |
|  |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Staff member invokes the Edit Order Function  **Step 4:** Staff member selects which order they would like to edit  **Step 6:** Staff member edits the order’s menu items | **Step2:** System retrieves the list of open orders from the **‘orders file’**  **Step 3:** System displays UI and awaits further input  **Step 5:** System prompts Staff member to edit the order’s menu items.  **Step 7:** System updates the order and sends updates to the chef if necessary  **Step 8:** Display confirmation message  **Step 8:** SystemUI is reset |
| **Alternate Scenarios** | **Manager** | **System** |
|  |  |  |
| **Conclusions** |  | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Cancel Order**

This function allows a staff member to edit an existing order in the till System.

Staff

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Cancel Order | |
| **Use Case Id** |  | |
| **Priority** | High | |
| **Source** | Staff | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the staff to cancel/void an existing order. | |
| **Preconditions** | The order the staff member wishes to cancel exists. | |
| **Trigger** |  | |
|  |  | |
| **Expected Scenario** | **Staff** | **System** |
|  | **Step 1:** Staff member invokes the Cancel Order Function  **Step 4:** Staff member selects which order they would like to cancel | **Step2:** System retrieves the list of open orders from the **‘orders file’**  **Step 3:** System displays UI and awaits further input  **Step 5:** System updates the order’s status to void (‘V’) in the **‘orders file’**  **Step 6:** SystemUI is reset |
| **Alternate Scenarios** | **Staff** | **System** |
|  |  |  |
| **Conclusions** | An order is cancelled and removed from the till. | |
| **Post conditions** | The order is no longer accessible in the till. | |
| **Business Rules** |  | |
| **Implementation Constraints** | If an order is cancelled without payment, the order must remain traceable for Revenue analysis/ stock counts etc. | |

### **Pay Bill**

This function allows a staff member to pay the bill on an existing order in the till System.

Staff

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Pay Bill | |
| **Use Case Id** |  | |
| **Priority** | High | |
| **Source** | Staff | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the staff to pay the bill on, and close, an order. | |
| **Preconditions** | The order the staff member wishes to pay the bill on exists. | |
| **Trigger** |  | |
|  |  | |
| **Expected Scenario** | **Staff** | **System** |
|  | **Step 1:** Staff member invokes the Pay Bill Function  **Step 3:** Staff member selects which order they would like to pay the bill on and close.  **Step 5:** Staff member selects which payment method was used. | **Step 2:** System displays UI and awaits further input  **Step 4:** System prompts which payment method was used:   * Cash * Card * Gift Card   **Step 6:** Systemupdates the order’s status to closed (‘C’) in the **‘orders file’**  **Step 7:** SystemUI is reset |
| **Alternate Scenarios** | **Staff** | **System** |
|  |  |  |
| **Conclusions** | An order is paid for and becomes unavailable | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** | After an order is paid for, the order must remain traceable for Revenue analysis/ stock counts etc. | |

## Perform Admin Functions

### **Item Analysis**

This function allows the Manager to generate financial information and best seller analysis on a till Item.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Item Analysis | |
| **Use Case Id** |  | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the manager to generate a revenue analysis on an item in the till System. | |
| **Preconditions** |  | |
| **Trigger** |  | |
|  |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Generate Item Analysis function.  **Step 3:** Manager selects which item they would like to generate a revenue analysis on. | **Step 2:** System retrieves the item’s sales data from the ‘**Sales-Info File’**  **Step 4:** The System generates a revenue analysis on the selected item.  **Step 5:** Systemdisplays the generated revenue analysis on the UI |
| **Alternate Scenarios** | **Manager** | **System** |
|  |  |  |
| **Conclusions** | Item revenue analysis is generated and displayed to the user. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Revenue Analysis**

This function allows the Manager to generate a yearly revenue analysis report.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Revenue Analysis | |
| **Use Case Id** |  | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the Manager to generate a yearly revenue analysis report. | |
| **Preconditions** |  | |
| **Trigger** |  | |
|  |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Generate Yearly Revenue Analysis Report function.  **Step 3:** Manager selects which year they would like to generate a revenue analysis on. | **Step 2:** System retrieves the yearly revenue data from the ‘**Revenue File’**  **Step 4:** The System generates a revenue analysis on the selected year.  **Step 5:** Systemdisplays the generated revenue analysis report on the UI. |
| **Alternate Scenarios** | **Manager** | **System** |
|  |  |  |
| **Conclusions** | Yearly revenue analysis is generated and displayed to the user. | |
| **Post conditions** | Yearly revenue is now available to view. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

# System Model

The following dataflow diagrams have been produced for the system:

DFD often do not contain members of staff included as they can be depicted as part of the System. However, as members of staff are the only entities interacting with the System processes, I have included them to better represent the data flow.

## Level-0 DFD

Place Orders

Menu Items

Restaurant Till System

Manager

Staff

Orders

Revenue Analysis

## Level-1 DFD

D3

Orders Menu Items File

D2

Orders File

Order Details

P2

Orders

Order Details

Staff

Order Details

P3

Admin

Menu Item Details

P1

Menu Items

Menu Item Details

D1

Menu Items file

Items Details

Admin Analysis

Manager

## Level-2 DFD (Process P1: Menu Items)

P1.4

Query Item

P1.3

Remove Item

Menu Item Availability Status

Items Details

New Item Details

Staff

D1

Menu Items file

Item Availability Status

P1.1

Add Item

Menu Items Details

New Menu Items Details

P1.2

Edit Item

Manager

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

Order Details

D3

Orders Menu Items File

Order Details

D1

Menu Items file

P2.1

Place Order

Staff

P2.4

Pay Bill

Order Status

Order Details

P2.3

Cancel Order

D2

Orders File

New Order Status

Order Details

New Order Details

Order Details

P2.2

Edit Order

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

Manager

Menu Items Details

D1

Menu Items file

Order Details

P3.2

Revenue Analysis

P3.1

Item Analysis

Menu Item Details

D3

OrderMenuItems file

Order Details

Order Details

D2

Orders File

# Data Model (Class Diagram)

This data model is comprised of a graphical UML class diagram, relational database schema, and a database schema for the Restaurant till System.

## Class Diagram

|  |
| --- |
| Order |
| OrderID: int (4)  OrderDate: date  OrderPrice: double (6, 2)  OrderStatus: char(1) |

|  |
| --- |
| MenuItem |
| ItemID: int (4) ItemType: char (1) ItemName: varChar (20) ItemDescription: varChar (20) ItemPrice: Number ItemStatus: char (1) |

1

1

has

has

|  |
| --- |
| OrderMenuItems |
| ItemID: int (4)  OrderID: int (4) |

1….\*

1….\*

## Relational Schema

Orders (OrderID, OrderDate, OrderPrice, OrderStatus)

OrderMenuItems (OrderID, ItemID,)

MenuItems (ItemID, ItemType, ItemName, ItemDescription, ItemPrice, ItemStatus)

## Database Schema

**Relation** Orders

**Attributes:**

OrderID: int (4) NOT NULL

OrderDate: date NOT NULL

OrderPrice: double (6, 2) NOT NULL

OrderStatus: char (1) NOT NULL

**Primary Key:** OrderID

**Relation** OrderMenuItems

**Attributes:**

OrderID: int (4) NOT NULL

ItemID: int (4) NOT NUL

**Foreign Key:** OrderID

**Foreign Key:** ItemID

**Relation** MenuItems

**Attributes:**

ItemID: int (4) NOT NULL

ItemType: char (1) NOT NULL CHECK ‘F’ OR ‘B’ OR ‘D’

ItemPrice: float (5,2) NOT NULL

**primary Key:** ItemID

# Conclusion

In this document I have outlined the Restaurant System’s operations and functions.

During the writing of this requirements paper I had difficulties with keeping track of information and how it would be correctly stored in the System without conflicts or unnecessary data storage wastage.

For the Restaurant System System's class diagram, I initially had a many to many or M:M relationship between **Orders** and **MenuItems.** However, this relationship would break the cardinality rules in database design, many-to-many relationships cannot be transformed into a relational database schema! Due to this I had to find a way to maintain the file’s entity integrity. To resolve this issue, I added the *Junction/ Weak Entity Type/ Relational link table* of **OrderMenuItems**.

Building the System prototype in Visual Studio parallel to the creation of this document was indispensable as it helped to troubleshoot any potential challenges that needed accounting for, and the prototype often highlighted logical errors I did not anticipate or could not visualize and therefore may have missed otherwise. I especially noticed the benefit when coding the remove order functionality in the prototype and realized that the Revenue Analysis functions were reliant on the information contained within the orders and menu items files. Similarly, an Order is comprised of menu items and therefore I had to consider the fact that a menu Item could be edited or even removed while it is in an order. This could cause critical dependency errors if not managed correctly in a real System.

# Appendices

## Appendix A – Menu Item Analysis

Chart, bar chart

Description automatically generated

## Appendix B – Yearly Revenue Analysis

Chart, histogram

Description automatically generated

## Appendix C – Declaration of Originality

Text

Description automatically generated