Requirements Engineering

Printer Ink Order System

Submitted By: Jack Patterson (t00217640)

Computing with Games Development

Date Submitted: 18/12/2021

**Table of Contents**

[1. Introduction/overview 4](#_Toc101645467)

[2. Functional Components 5](#_Toc101645468)

[3. User Requirements 6](#_Toc101645469)

[3.1. OrderSYS will manage Suppliers 6](#_Toc101645470)

[3.2. OrderSYS will manage products 6](#_Toc101645471)

[3.3. OrderSYS will manage orders 6](#_Toc101645472)

[3.4. OrderSYS will perform administration 6](#_Toc101645473)

[4. System Requirements 7](#_Toc101645474)

[4.1. System Level Use Case Diagram 7](#_Toc101645475)

[4.2. Manage Supplier 8](#_Toc101645476)

[4.2.1 Register Supplier 8](#_Toc101645477)

[4.2.2 Update Supplier 11](#_Toc101645478)

[4.3 Manage Products 14](#_Toc101645479)

[4.3.1 Add Product 14](#_Toc101645480)

[4.3.2 Update Product 16](#_Toc101645481)

[4.3.3 Discontinue Product 18](#_Toc101645482)

[4.4 Manage Orders 20](#_Toc101645483)

[4.4.1 Place Order 20](#_Toc101645484)

[4.4.2 Pay Invoice 23](#_Toc101645485)

[4.4.3 List Outstanding Payments 25](#_Toc101645486)

[4.5 Perform Admin 27](#_Toc101645487)

[4.5.1 Perform Yearly Revenue Analysis 27](#_Toc101645488)

[4.5.2 Perform Yearly Supplier Analysis. 28](#_Toc101645489)

[5. System Model 30](#_Toc101645490)

[5.1. Level-0 DFD 30](#_Toc101645491)

[5.2. Level-1 DFD 31](#_Toc101645492)

[5.3. Level-2 DFD (Process P1: Manage Supplier) 32](#_Toc101645493)

[5.4. Level-2 DFD (Process P2: Manage Products) 33](#_Toc101645494)

[5.5. Level-2 DFD (Process P3: Manage Orders) 34](#_Toc101645495)

[5.6. Level-2 DFD (Process P4: Perform Admin) 35](#_Toc101645496)

[6. Data Model (Class Diagram) 36](#_Toc101645497)

[6.1. Class Diagram 36](#_Toc101645498)

[6.2. Relational Schema 37](#_Toc101645499)

[6.3. Database Schema 37](#_Toc101645500)

[7. Conclusion 39](#_Toc101645501)

[8. Appendices 40](#_Toc101645502)

[8.1. N/A 40](#_Toc101645503)

[8.2. N/A 40](#_Toc101645504)

# Introduction/overview

The OrderSys is made up of four core parts which respectively manage suppliers, products, orders as well as perform admin. Each of them is further broken down into smaller processes which carry out a certain task. For example, there is a process to add a supplier and there is one to update a supplier.

Each of them communicates with a database made up of four files. Each table contains core information necessary for the system to work as intended. This document is laid out with multiple diagrams and models in order to allow the system to be made.

# Functional Components

# User Requirements

## OrderSYS will manage Suppliers

* + 1. OrderSYS will register a supplier
    2. OrderSYS will update a supplier

## OrderSYS will manage products

* + 1. OrderSYS will add a product
    2. OrderSYS will update a product
    3. OrderSYS will discontinue a product

## OrderSYS will manage orders

* + 1. OrderSYS will place an order
    2. OrderSYS will record a supplier payment
    3. OrderSYS will list outstanding payments

## OrderSYS will perform administration

* + 1. OrderSYS will produce a yearly revenue analysis
    2. OrderSYS will provide a yearly supplier analysis

# System Requirements

## System Level Use Case Diagram

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

Manager

Supplier

## Manage Supplier

### Register Supplier

This function registers the details of a supplier. A supplier is assigned a unique SuppID.

Manager

<<includes>>

<<executes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Register Supplier** | |
| **Use Case Id** | OS001 | |
| **Priority** | 1 | |
| **Source** | Business Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Supplier | |
| **Description** | This function registers a supplier from another business into the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System** |
|  | **Step 1:** Manager invokes the Register Supplier Function.  **Step 4:** Supplier enters the required data:   * Name * Address (Eir Code, Street, Town, County) * Email * PhoneNo * Contact | **Step 2:** The System generates the SuppID.  **Step 3:** The System displays the UI.  **Step 5:** The system checks if the data is valid.   * Verifies if address Eir code is in the correct format. * Verifies that the county is in the list of valid counties. * Verifies if the email is a proper email format. * Verifies that the PhoneNo is in the correct format.   **Step 6:** The system saves the data in the suppliers file:   * SuppID * Name * Address (Eir Code, Street, Town, County) * Email * PhoneNo * Contact   **Step 7:** Display confirmation message  **Step 8:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data** |  | **Step 5:** The system checks if the data is valid.   * Verifies that the county is in the list of valid counties. * Verifies if address Eir code is in the correct format. * Verifies if the email is a proper email format. * Verifies that the PhoneNo is in the correct format.   **Step 6:** The system displays an error message stating that the validation failed.  **Step 7:** The system returns to Step 4. |
| **Conclusions** | The Supplier has been added to the database. | |
| **Post conditions** | The business can now be referenced in the system and its info can be used in the database. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update Supplier

This function updates the details of a supplier. It uses the unique SuppID to identify suppliers.

Manager

<<includes>>

<<executes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Supplier** | |
| **Use Case Id** | OS002 | |
| **Priority** | 1 | |
| **Source** | Business Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Supplier | |
| **Description** | This function updates an existing supplier from another business into the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System** |
|  | **Step 1:** Manager invokes the Update Supplier Function.  **Step 4:** The manager enters supplier name (or part of).  **Step 6:** The manager selects which supplier to update.  **Step 9:** The manager enters the required updated data.   * Name * Address (Eir Code, Street, Town, County) * Email * PhoneNo * Contact | **Step 2:** The System displays the UI.  **Step 5:** The System retrieves a summary of all suppliers with a matching name from the supplier file and loads it on the UI.  **Step 7:** The System retrieves the supplier data of the selected supplier from the supplier file:   * Name * Address (Eir Code, Street, Town, County) * Email * PhoneNo * Contact   **Step 8:** The System displays the UI.  **Step 10:** The system checks if the data is valid.   * Verifies if address Eir code is in the correct format. * Verifies that the county is in the list of valid counties. * Verifies if the email is a proper email format. * Verifies that the PhoneNo is in the correct format.   **Step 11:** The system updates the data in the suppliers file:   * Company Name * Address (Eir Code, Street, Town, County) * Email * PhoneNo * Contact   **Step 12:** Display confirmation message  **Step 13:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data** |  | **Step 10:** The system checks if the data is valid.   * Verifies that the Company Name begins with a two letter prefix followed by an underscore. * Verifies if address Eir code is in the correct format. * Verifies if the email is a proper email format. * Verifies that the PhoneNo is in the correct format.   **Step 11:** The system displays an error message stating that the validation failed.  **Step 12:** Return to Step 9. |
| **Conclusions** | The Supplier has been updated in the database. | |
| **Post conditions** | The business can now be referenced in the system and its updated info can be used in the database. | |
| **Business Rules** | Only existing suppliers can be updated. | |
| **Implementation Constraints** |  | |

## Manage Products

### Add Product

This function registers the details of a product. It uses the unique ProductID. It uses the unique SuppID to associate itself with a supplier. Quantity will be 0 until products are bought.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Product** | |
| **Use Case Id** | OS003 | |
| **Priority** | 1 | |
| **Source** | Business Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function adds a product owned by a business to sell. This determines the cost of the product. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Add Product Function.  **Step 5:** Manager enters the required data:   * SuppID * ProdID * Name * Qty * Price * Status | **Step 2:** The System retrieves SuppID and Name for all suppliers in order of supplier name from Supplier file and loads on the UI.  **Step 3:** The system generates a prodID.  **Step 4:** The System displays the UI.  **Step 6:** The system checks if the data is valid:   * All fields must be entered * Name must not be numeric. * Price must be numeric and greater than 0. * Qty (Zero)   **Step 7:** Set status to Available (‘A’)  **Step 8:** The system saves the data in the products file:   * ProductID * Name * Price * Qty * SuppID * Status   **Step 9:** Display confirmation message  **Step 10:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data** |  | **Step 6:** The system checks if the data is valid:   * All fields must be entered * Name must not be numeric and must begin with XX\_. (eg HP\_345) * Price must be numeric and greater than 0 * Qty (Zero)   **Step 7:** The system displays an error message stating that the validation failed.  **Step 8:** Return to Step 5. |
| **Conclusions** | The product has been added in the database. | |
| **Post conditions** | The product can now be referenced in the system and its info can be used in the database. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update Product

This function updates the details of a product. It uses the unique ProductID to identify products.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Product** | |
| **Use Case Id** | OS004 | |
| **Priority** | 1 | |
| **Source** | Business Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function updates a product owned by a business to sell. This determines the new cost of the product. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Update Product Function.  **Step 3:** The manager selects which product to update.  **Step 5:** Manager enters the required data:   * SuppID * ProdID * Name * Qty * Price * Status | **Step 2:** The system retrieves the product name from the database and displays on the UI.  **Step 4:** The system retrieves the product data for the selected product from the products table and displays on the UI.  **Step 6:** The system checks if the data is valid:   * All fields must be entered * Name must not be numeric and must begin with XX\_. (eg HP\_345) * Price must be numeric and greater than 0 * Qty (Zero)   **Step 7:** Set status to Available (‘A’).  **Step 8:** The system updates the data in the products file:   * ProductID * Name * Price * Qty * Status * SuppID   **Step 9:** Display confirmation message  **Step 10:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  | **Step 6:** The system checks if the data is valid:   * All fields must be entered * Name must not be numeric and must begin with XX\_. (eg HP\_345) * Price must be numeric and greater than 0 * Qty (Zero)   **Step 7:** The system displays an error message stating that the validation failed.  **Step 8:** The system will switch focus to the place with the error.  **Step 9:** Return to Step 5. |
| **Conclusions** | The product has been updated in the database. | |
| **Post conditions** | The product can now be referenced in the system and its updated info can be used in the database. | |
| **Business Rules** | Only existing products can be updated. | |
| **Implementation Constraints** |  | |

### Discontinue Product

This function is used to discontinue products. The product uses the unique ProductID to identify products.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Discontinue Product** | |
| **Use Case Id** | OS005 | |
| **Priority** | 1 | |
| **Source** | Business Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function discontinues a product owned by a business to sell. This removes the selected product from the database. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Discontinue Product Function.  **Step 3:** The manager selects which suppliers to have their product discontinued.  **Step 5:** The manager selects which product to discontinue.  **Step 8:** Manager confirms that they wish to discontinue the product. | **Step 2:** The system retrieves the Supplier name and SuppID from the Products file and displays on the UI.  **Step 4:** The system retrieves the product name and ProdID from the Products file which have a matching SuppID and are not already discontinued and displays on the UI.  **Step 6:** The system retrieves the product data for the selected product from the products file and displays on the UI.  **Step 9:** Set status to Discontinued (‘D’).  **Step 10:** Display confirmation message.  **Step 11:** Reset UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The product has been discontinued. | |
| **Post conditions** |  | |
| **Business Rules** | Only existing products can be discontinued. | |
| **Implementation Constraints** |  | |

## Manage Orders

### Place Order

This function records the details of an order sent to a supplier. An order is assigned a unique OrderID. An order can consist of more than one product.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Place Order** | |
| **Use Case Id** | OS006 | |
| **Priority** | 1 | |
| **Source** | Business Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function places an order from a business to buy a product to sell. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Place Order Function.  **Step 5:** Select required supplier.  **Step 7:** Manager enters the required data:   * Product * Quantity | **Step 2:** Assign next OrderID.  **Step 3:** The System retrieves SuppID and Name for all suppliers from the Supplier file in order of Name.  **Step 4:** Display UI.  **Step 6:** The system retrieves the available products data from the Products file for the selected supplier.  **Step 8:** The system checks if the data is valid.   * Quantity must be greater than 0. * Must verify there is sufficient stock to meet Quantity.   **Step 9:** Calculate cost of items.  **Step 10:** Add product details to shopping cart:   * ProdID (4 digits) * Quantity (3 digits) * Sale Price (7 digits) * Name   **Step 11:** Update shopping cart total.  **Step 12:** If more items to be ordered return to Step 7.  **Step 13:** The system saves the order details in the Orders file:   * OrderID * Order Date * Order Value * SuppID   **Step 14:**  For each item in the cart the system saves each order item in the OrderItems file:   * OrderID * ProdID * Quantity * Sale Price   **Step 15:** Display confirmation message.  **Step 16:** Reset UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  | **Step 1:** Manager invokes the Place Order Function.  **Step 4:** Select required supplier  **Step 5:** Manager enters the required data:   * Product(s) * Amount(s) * Date of order | **Step 2:** The System retrieves SuppID and Name for all suppliers from the Supplier file in order of Name  **Step 3:** Display UI  **Step 5:** System retrieves full details for selected supplier from Supplier file and loads on UI for viewing.  **Step 3:** The system retrieves the product data from the database  **Step 4:** The System displays the UI.  **Step 6:** The system checks if the data is valid.   * Verifies that at least one product has been selected. * Verifies that all products selected have come from the one supplier. * Verifies that at least one of each product selected was ordered. * Verifies that the date of order is the current day. * Verifies that the date of delivery is at least the current day.   **Step 7:** Display error message.  **Step 10:** Reset all invalid data on UI. |
| **Conclusions** | The product has been ordered from the business. | |
| **Post conditions** |  | |
| **Business Rules** | An order can only be placed by a single business. | |
| **Implementation Constraints** |  | |

### Pay Invoice

This function is used to pay invoices. An invoice uses the OrderID to identify an order. It will consist of only one order.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Pay Invoice** | |
| **Use Case Id** | OS007 | |
| **Priority** | 1 | |
| **Source** | Business Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function pays an outstanding invoice. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Pay Invoice Function.  **Step 4:** Select the supplier to pay.  **Step 6:** Manager selects the the invoice they wish to pay. | **Step 2:** The System retrieves the supplier data from the supplier file.  **Step 3:** The System displays the UI.  **Step 5:** System retrieves name and suppID of the supplier along with their order details for any orders where InvPaid = ‘N’ and loads on UI.  **Step 7:** The system marks the invoice as paid (‘Y’).  **Step 8:** Display confirmation message.  **Step 9:** Reset UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The invoice has been marked as paid and will no longer be listed as outstanding. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### List Outstanding Payments

This function lists all outstanding payments. It uses the OrderID to identify an order. It may list multiple outstanding orders.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **List Outstanding Payments** | |
| **Use Case Id** | OS008 | |
| **Priority** | 1 | |
| **Source** | Business Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function lists outstanding payments. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the List Outstanding Payments Function.  **Step 4:** Manager searches for and selects which supplier to list payments for.  **Step 6:** The manager may press a button to reset the UI and reselect a new supplier to view. | **Step 2:** The System retrieves the supplier SuppID and Name from the supplier file.  **Step 3:** The System displays the UI.  **Step 5:** The system retrieves the order data and orderitem data for the supplier and displays it on the UI.  **Step 7:** Reset UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The outstanding payments are listed. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Perform Admin

### Perform Yearly Revenue Analysis

This function performs a yearly revenue analysis. It uses the unique OrderID to collect the orders for the year. It can consist of many orders.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Yearly Revenue Analysis** | |
| **Use Case Id** | OS009 | |
| **Priority** | 1 | |
| **Source** | Business Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function displays an analysis of revenue of the last year. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Yearly Revenue Analysis Function.  **Step 3:** The manager chooses which year to view revenue for. | **Step 2:** The System displays the UI.  **Step 4:** The System retrieves the order data for that year and organises it by month and displays on the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The Yearly Revenue Analysis is shown. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Perform Yearly Supplier Analysis.

The function performs a yearly supplier analysis. It uses the unique OrderID and SuppID to collect the orders for the year. It may consist of many orders.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Yearly Supplier Analysis** | |
| **Use Case Id** | OS010 | |
| **Priority** | 1 | |
| **Source** | Business Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function displays an analysis of suppliers of the last year. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Manager invokes the Yearly Revenue Analysis Function.  **Step 3:** The manager chooses which year to view revenue for and also selects which supplier to choose revenue for. | **Step 2:** The System displays the UI.  **Step 4:** The System retrieves the order data for that year and organises it by month for that supplier by selecting months which have that SuppID and displays on the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The Yearly Supplier Analysis is shown. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

Products

OrderSys

Process

Jobs

Order details

Supplier

## Level-1 DFD

D4

Order Items File

D3

Order File

Product Details

Product Details

Order Item Details

Order Details

Order Details

Supplier Details

Supplier Details

P4

Manage Products

D2

Product File

D1

Supplier File

P3

Manage Orders

P4

Perform Admin

P1

Manage Supplier

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

Supplier Details

P1.2

Update Supplier

P1.1

Register Supplier

Supplier Details

Supplier Details

D1

Supplier File

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

Product Details

P2.2

Update Product

P2.1

Add Product

Supplier Details

Supplier Details

Supplier Details

D1

Supplier File

Product Details

Product Details

Product Details

Remove Product Details

P2.3

Dincontinue Product

D2

Product File

## Level-2 DFD (Process P3: Manage Orders)

Order Details

P3.1

Place Order

P3.2

Pay Invoice

D3

Order File

Invoice

Order Details

D1

Supplier File

Order Details

Order Details

P3.3

List Outstanding Payments

D4

Order Item File

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

D1

Supplier File

D2

Product File

D3

Order File

P4.1

Perform Yearly Revenue Analysis

P4.2

Perform Yearly Supplier Analysis

# Data Model (Class Diagram)

## Class Diagram

Order

- orderID : int\*

- orderDate : calendar

- total: decimal

- suppID: int\*

- invPaid: char

+ getNextOrdID()

+ removeQuantity()

+ getQuantity()

+ getPrice()

+ placeOrder()

+ getUnpaidOrder()

+ getSuppData()

+ getOrdItemData()

+ getProdName()

+ getTotal()

+ payInvoice()

+ getDate()

Product

- prodID : int\*

- suppID : int\*

- name : String

- qty : int

- status : char

- price : float

+ registerProduct()

+ getNextProdID()

+ getID()

+ getProdID()

+ setSelectedItem()

+ searchAllProdName()

+ searchAllProdInfo()

+ discontinueProd()

+ updateProd()

+ searchProdQty()

Supplier

- suppID : int\*

- name : String

- street : String

- town : String

- county : String

- eircode : String

- phoneNo : int

- contact : String

+ registerSupplier()

+ getNextSuppID()

+ searchSuppName()

+ searchAllSuppName()

+ searchAllSuppInfo()

+ setSelectedItem()

+ updateSupplier()

Admin

+ getYearlyRevenue()

+ getYearlySuppRevenue()

OrderItem

- orderID : int\*

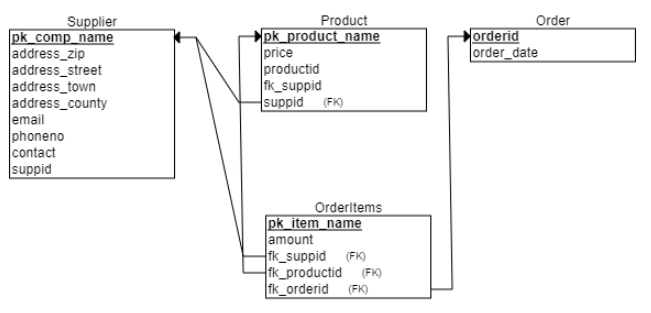
- prodID : int\*

- qty: int

- salePrice: decimal

+ addOrderItem

## Relational Schema

******

## Database Schema

**Schema:** OrderSys

**Relation** *Suppliers*

SuppID NUMERIC(5) NOT NULL, UNIQUE

Name VARCHAR(30) NOT NULL, UNIQUE

Address\_Street VARCHAR(20) NOT NULL

Address\_Town VARCHAR(20) NOT NULL

Address\_County VARCHAR(20) NOT NULL

Address\_Eir VARCHAR(7)

Email VARCHAR(20) NOT NULL, UNIQUE

PhoneNo NUMERIC(13) NOT NULL

Contact VARCHAR(30)

**Constraint:** pk\_suppliers **Primary Key:** SuppID

**Relation** *Products*

ProdID NUMERIC(4) NOT NULL, UNIQUE

Name VARCHAR(20) NOT NULL, UNIQUE

Price NUMERIC(6,2) NOT NULL

Qty NUMERIC(4) NOT NULL

Status CHAR NOT NULL

SuppID NUMERIC(4)

**Constraint:** pk\_products **Primary Key:** ProdID

**Constraint:** fk\_suppliers **Foreign Key:** SuppID **References** Supplier

**Relation** *Order*

OrderID NUMERIC(4)

OrderDate DATE

OrderValue NUMERIC (6,2)

SuppID NUMERIC (4)

PaidInv CHAR(1)

**Primary Key:** OrderID

**Relation** *OrderItems*

OrderID NUMERIC(4)

ProdID NUMERIC (4)

Quantity NUMERIC(3)

Price NUMERIC (6,2)

**Primary Key:** OrderID,ProdID

**[Foreign Key:** OrderID **References** Order]

**[Foreign Key:** SuppID **References** Supplier]

**[Foreign Key:** ProductID **References** Product]

# Conclusion

In order to create a completed system ten sub-processes under four main processes are needed which manage Suppliers, Products, Orders and Administration in order to work. They require four database tables to work as intended.

# Appendices

## N/A

## N/A