**Introduction to Project**

Mr. Smith is the owner of a store, selling different products (e.g., food, fruit, toy). He is currently managing on paper. We want to sell him a store management system to handle these tasks.

The main business process of the store is the sale of items. In order to facilitate this, inventory must be managed, and products must be purchased from suppliers. These processes involve a variety of people who both work at the store and interact with those who do. The people involved in all of these processes can be grouped into 5 main roles: customer, cashier, stocker, manager, and supplier. Mr. Smith falls into the category of manager. The responsibilities of each role are based on the activities required in the processes introduced above

To get a better picture of what the Store Management System must do, three business processes can be divided into their tasks and who is responsible for completing them:

1. Sell items to customers:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | Customer | Cashier | Stocker | Manager | Supplier |
| Present items for purchase | RESPONSIBLE |  |  |  |  |
| Check item cost and calculate total |  | RESPONSIBLE |  |  |  |
| Provide payment | RESPONSIBLE |  |  |  |  |
| Provide receipt |  | RESPONSIBLE |  |  |  |

2. Manage shop inventory

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | Customer | Cashier | Stocker | Manager | Supplier |
| Remove items from shelves | RESPONSIBLE |  |  |  |  |
| Find empty shelves |  |  | RESPONSIBLE |  |  |
| Restock with correct item |  |  | RESPONSIBLE |  |  |
| Log items removed from inventory |  |  | RESPONSIBLE |  |  |

3. Purchase new items from suppliers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | Customer | Cashier | Stocker | Manager | Supplier |
| Identify desired products for store |  |  |  | RESPONSIBLE |  |
| Provide item and total costs |  |  |  |  | RESPONSIBLE |
| Pay for new items |  |  |  | RESPONSIBLE |  |
| Provide receipt and ship items |  |  |  |  | RESPONSIBLE |
| Replace inventory at store |  |  | RESPONSIBLE |  |  |

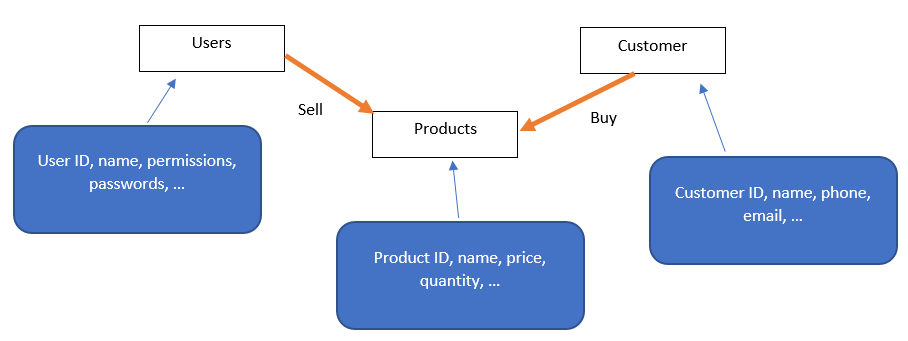
It is not possible for the Store Management System to completely replace all of these processes, so we must draw an entity relationship diagram to determine which entities can be handled by the system.

Entities can be grouped into the following example sets

* Users:
  + Example entities: Mr. Smith, cashiers,…
  + Stored information: Working hours, name, employee ID, photo, permissions, passwords
* Products:
  + Example entities: food, toys, clothing
  + Stored information: Barcode, name, product ID, expiration date, price, tax, amount, supplier, date received
* Customers:
  + Example entities: in-store, online
  + Stored information: name, address, customer ID, phone number, email, credit card
* Suppliers:
  + Stored information: name, address, products purchased, supplier ID, cost

The following relationships can be made between the sets:

* Customers purchase products
* Users at the store interact with the products during stocking
* Users also interact with customers during sale
* Some users may purchase new items from suppliers



Based on the information gathered above, we can begin design of the system. The system is to be broken into 3 main tasks: Add Product, Add Customer, and Add Purchase.

The tasks outlined in the project are completed below

**Task 1**: Write two possible use cases for each user story: one is the common case and one is the exception.

Example users: Customer, Cashier, Stocker, Manager, Supplier

User Story 1: As a user, I want to add a new product into the system.

1. As a stock clerk, I want to add newly received inventory items to the product database
2. As a manager, I want to add a product without a quantity

User Story 2: As a user, I want to add a new customer into the system.

1. As a cashier, I want to add a new customer to the database during checkout
2. As a manager, I want to edit the information of an existing customer

User Story 3: As a user, I want to add a purchase from a customer into the system.

1. As a cashier, I want to add a purchase for an existing customer buying existing products
2. As a manager, I want to add a purchase with a new product/customer

Each of the “exception cases” is beyond the scope of the Store Management System as outlined by the requirements of this project thus far. Therefore, an attempt to perform these actions should result in an error case that prevents the action from being performed without damaging functionality or stored data.

* User Story 1, Case 2: Quantity must be a number and cannot be blank
* User Story 2, Case 2: Only adding new customers is allowed, and customer ID must be unique
* User Story 3, Case 2: Products and customers must exist in their tables before being applied to a purchase

Functionality is tested by demonstrating typical use for each of the 3 functions, and by showing all possible error cases for each of the 3 functions. Demonstration of the error cases encapsulates all “exception” cases

Video documentation of testing: <https://www.youtube.com/playlist?list=PLxSttXKfgYWRzLSG_KqQxz8F5VNCRh9ui>

**Task 2**: Design the screens (UI windows and widgets) the system should display in each use case.

Add Product menu – Normal Use

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. Choose “Add Product” on main screen | 2. Display “Add Product” screen |
| 3. Input data then click “Add” button | 4. Display “Product Added Successfully” screen with product database information |
| 5. Click “Ok” on previous screen | 6. Display “Add Product” screen again |
| 7. Click “Cancel” on Add Product screen | 8. Display “Product Add Cancelled” screen |
| 9. Click “Ok” on previous screen | 10. Display main screen again |

There are constraints on inputs to the Add Product screen. When invalid inputs are entered, an error screen displays

* Product ID:
  + Cannot be null
  + Must be unique
  + Must be an integer
* Name:
  + Cannot be empty
* Price and Quantity:
  + Must be numbers

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. Choose “Add Product” on main screen | 2. Display “Add Product” screen |
| 3. Enter data which does not meet the criteria above and click “Add” | 4. Display Error Message screen    Possible error messages:   * ProductID cannot be null * ProductID is invalid * Product name cannot be empty * Price is invalid * Quantity is invalid * Duplicate Product ID, product not added |
| 5. Click “Ok” on previous screen | 6. Display previous Add Product screen with information still visible for user to correct |

Add Customer menu – Normal Use

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. Choose “Add Customer” on main screen | 2. Display “Add Customer” screen |
| 3. Input data then click “Add” button | 4. Display “Customer Added Successfully” screen with customer database information |
| 5. Click “Ok” on previous screen | 6. Display “Add Customer” screen again |
| 7. Click “Cancel” on Add Customer screen | 8. Display “Customer Add Cancelled” screen |
| 9. Click “Ok” on previous screen | 10. Display main screen again |

There are constraints on inputs to the Add Customer screen. When invalid inputs are entered, an error screen displays

* Customer ID:
  + Cannot be null
  + Must be unique
  + Must be an integer
* Name:
  + Cannot be empty
* Phone:
  + Must be a number

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. Choose “Add Customer” on main screen | 2. Display “Add Customer” screen |
| 3. Enter data which does not meet the criteria above and click “Add” | 4. Display Error Message screen    Possible error messages:   * CustomerID cannot be null * CustomerID is invalid * Customer name cannot be empty * Phone is invalid * Duplicate Customer ID, customer not added |
| 5. Click “Ok” on previous screen | 6. Display previous Add Customer screen with information still visible for user to correct |

Add Purchase menu – Normal Use

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. Choose “Add Purchase” on main screen | 2. Display “Add Purchase” screen |
| 3. Input data then click “Add” button | 4. Display “Purchase Added Successfully” screen with receipt for purchase |
| 5. Click “Ok” on previous screen | 6. Display “Add Purchase” screen again |
| 7. Click “Cancel” on Add Purchase screen | 8. Display “Purchase Add Cancelled” screen |
| 9. Click “Ok” on previous screen | 10. Display main screen again |

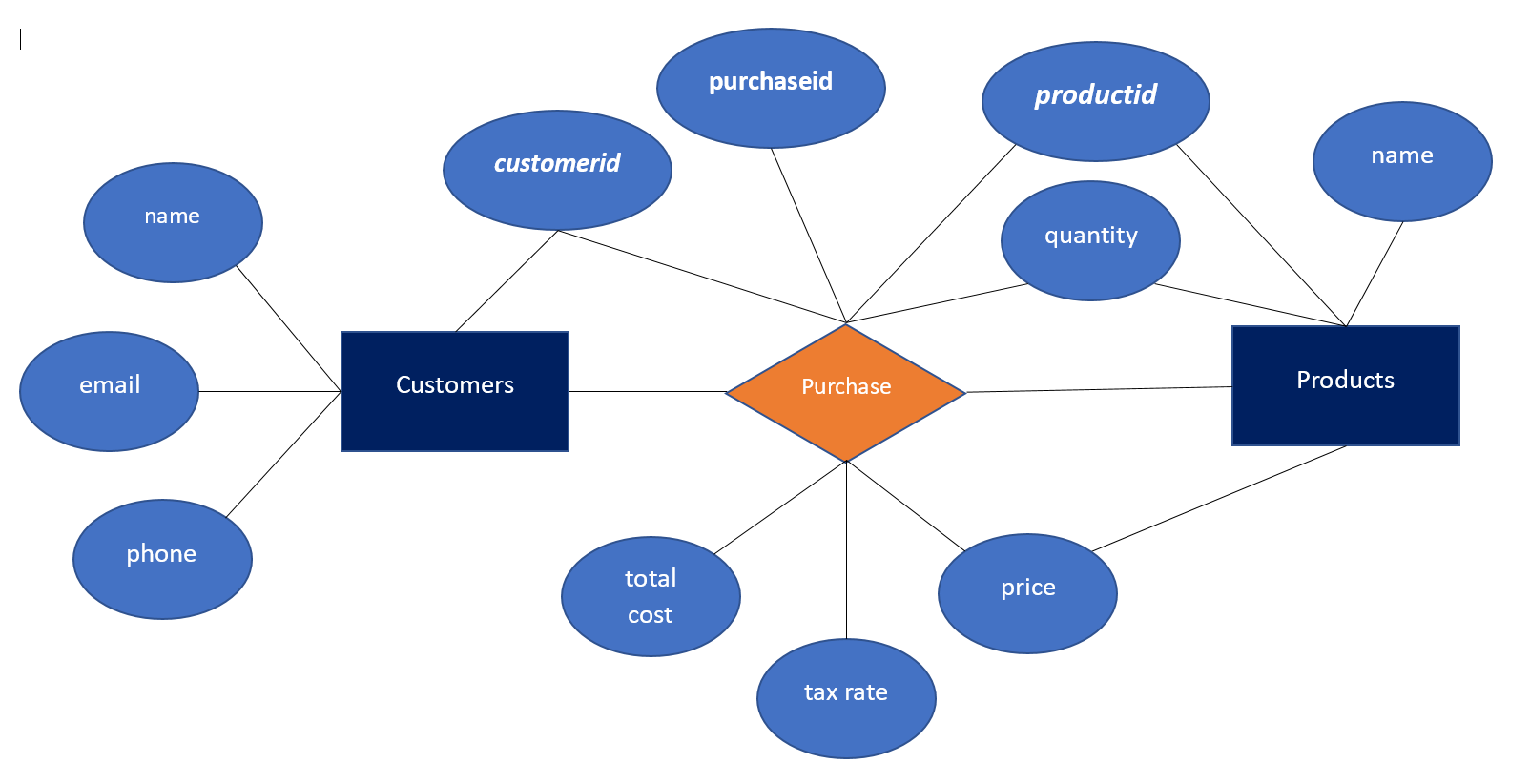
There are constraints on inputs to the Add Purchase screen. When invalid inputs are entered, an error screen displays

* Purchase ID:
  + Cannot be null
  + Must be unique
  + Must be an integer
* Product ID and Customer ID
  + Cannot be null
  + Must be an integer
  + Must match existing value in database
* Quantity:
  + Must be an integer

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. Choose “Add Purchase” on main screen | 2. Display “Add Purchase” screen |
| 3. Enter data which does not meet the criteria above and click “Add” | 4. Display Error Message screen    Possible error messages:   * PurchaseID cannot be null * PurchaseID is invalid * ProductID cannot be null * ProductID is invalid * ProductID does not exist * CustomerID cannot be null * CustomerID is invalid * CustomerID does not exist * Quantity is invalid * Duplicate Purchase ID, purchase not added |
| 5. Click “Ok” on previous screen | 6. Display previous Add Purchase screen with information still visible for user to correct |

**Task 3**: Design the database physically and prepare data for the tables, with at least 5 products, 5 customers, and 10 purchases.

The following relationship diagram is the basis for the database design. When a purchase is made, the ProductID, CustomerID, and Quantity for existing table values are passed into the Purchase.



Based on the above diagram, database tables are created using the SQL queries on the left, and preliminary data is entered in the tables using the queries on the right.

|  |  |
| --- | --- |
| CREATE TABLE "Customers" (  "CustomerId" INTEGER NOT NULL,  "Name" TEXT DEFAULT 'Guest',  "Phone" INTEGER,  "Email" TEXT,  PRIMARY KEY("customerid")  ); | INSERT INTO Customers  VALUES  (1,'Jane', 1234567890, 'jane@email.com'),  (2,'Emily', 1029384756, 'emily@email.com'),  (3,'Ryan', 9876543210, 'ryan@email.com'),  (4,'Katy', 3216540987, 'katy@email.com'),  (5,'John', 3214325432, 'john@email.com'); |
| CREATE TABLE "Products" (  "ProductId" INTEGER NOT NULL,  "Name" TEXT,  "Price" REAL,  "Quantity" REAL DEFAULT 1,  PRIMARY KEY("ProductID")  ); | INSERT INTO Products  VALUES  (1, 'Apple', 0.99, 1),  (2, 'Chicken Nuggets', 7.99, 12),  (3, 'iPhone', 1299.99, 1),  (4, 'Hockey Stick', 230.00, 1),  (5, 'Oranges', 4.00, 3); |
| CREATE TABLE "Purchases" (  "PurchaseId" INTEGER NOT NULL,  "ProductId" INTEGER NOT NULL,  "CustomerId" INTEGER NOT NULL,  "Quantity" INTEGER DEFAULT 1,  "Price" REAL,  "Tax" REAL,  "TotalCost" REAL,  PRIMARY KEY("PurchaseId")  ); | INSERT INTO Purchases  VALUES  (1, 1, 1, 10, 9.90, 0.79, 10.69),  (2, 5, 1, 3, 12.00, 0.96, 12.96),  (3, 2, 2, 3, 23.97, 1.92, 25.89),  (4, 3, 2, 1, 1299.99, 104.00, 1403.99),  (5, 4, 3, 1, 230.00, 18.40, 248.40),  (6, 5, 3, 1, 4.00, 0.32, 4.32),  (7, 2, 4, 2, 15.98, 1.28, 17.26),  (8, 1, 4, 5, 4.95, 0.40, 5.35),  (9, 4, 5, 2, 460.00, 36.80, 496.80),  (10, 3, 5, 1, 1299.99, 104.00, 1403.99); |

**Task 4**: Implement the use cases.

The use cases are implemented using a 3-tier system, consisting of:

1. User Interface
2. Business Logic
3. Data access

User Interface:

This is what the user interacts with, and consists of 4 primary views (as outlined above):

* MainView class
* AddProductView class
* AddCustomerView class
* AddPurchaseView class
* StoreManager class: main class to run the store manager

Business logic:

This is where information entered by a user is interpreted and evaluated in terms of the goals of the store management system. Models for the classes allow for information to be passed into a database:

* ProductModel class: store information of a product
* CustomerModel class: store information of a customer
* PurchaseModel class: store information of a purchase

Data access:

This is where information accumulated in the business logic tier is stored. The tier includes the adapter to connect with a database, as well as the methods to save and store information in that database

* IDataAdapter class:
* SQLiteAdapter: connect to SQL database
* OracleDataAdapter class: connect to Oracle database (for future implementation)

The above classes are implemented in Java

**Task 5**: Test the system with each use case.

Functionality is tested by demonstrating typical use for each of the 3 functions, and by showing all possible error cases for each of the 3 functions. Demonstration of the error cases encapsulates all “exception” cases

Video documentation of testing: <https://www.youtube.com/playlist?list=PLxSttXKfgYWRzLSG_KqQxz8F5VNCRh9ui>