**Patel Vrundaben Vijaykumar-158605220**

Inventory Module

Linda buy products from local crafters, designers and bakers. One of her frequent customers has been diagnosed with celiac disease and has approached her to include gluten free products.

As well, Linda wonders about food recalls and how would they be able to inform her customers if they have purchased a recalled product.

Linda would like a system that helps her organize her purchases by location, instead of having to look through a combination of online orders and paper receipts. She would like the ability to attach a copy or picture of the receipt to the purchase for tax purposes.

Please create a class diagram and sequence diagrams for the User Stories and Systems Use Case Specifications detailed below.

**Copy and paste your work into a MSWord compatible file and include the code that Visual Paradigm would output from your model.**

Use Case: Maintain Products

User Story

As the owner of this business, I would like to record products so that I can easily see what I’ve purchased, what I’ve sold and how much inventory that I have at each location.

Acceptance Criteria:

1. Must be able to record critical information about each product.
2. Allow deletion of product, in case the owner makes a mistake. Owner must be prompted to confirm and product must not be involved in any purchases or sales.
3. Must be able to easily retrieve product information.
4. Must be able to retrieve a list of products by product type.

A diagram of a product

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Create Product Information | | |
| Triggering Event | A new product of interest to the business | | |
| Brief Description | Allows the Owner to record a new product. | | |
| Actors | Owner | | |
| Related Use Cases |  | | |
| Preconditions | Owner has opened the Main Menu. | | |
| Post Conditions | Product is saved to the database and now can be purchased. | | |
| Flow of activities | Actor | | System |
|  |  | Requests to add a new product | Displays a list of product types and prompts for selection.  Prompts for product name, ingredients, description, price |
|  |  | Enters name, description, ingredients, and instructions for use  Selects product type | name, description and price must be entered.  product type must be selected  Generates identifier  Data is valid  Displays product  Prompts to save |
|  |  | Request to save | Saves the product and returns to the main menu |
| Exception Conditions | * Owner chooses to cancel adding the product | | |

A diagram of a computer program

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Query Products by Product Type | | |
| Triggering Event | List of products required | | |
| Brief Description | Allows the Owner to query products | | |
| Actors | Owner | | |
| Related Use Cases |  | | |
| Preconditions | Owner has opened the Main Menu. | | |
| Post Conditions | Product information provided to the actor. | | |
| Flow of activities | Actor | | System |
|  | 1. | Requests to products | Displays a list of product types and prompts for selection |
|  | 2. | Selects product type | Displays a list of products for that product type |
| Exception Conditions | * Owner chooses to cancel query products | | |

A diagram of a product

Description automatically generated

Use Case: Maintain Purchase

User Story

As the owner of this business, I would like to record my purchases so that I can quickly see how much money I’ve spent, what I’ve ordered and from where. With each purchase, I want to record which location received the product. Because I like to order in bulk, I must be able to record multiple locations for each product.

Acceptance Criteria:

1. Must be able to record purchases by supplier.
2. Must be able to query purchase details by location.

Use Case Descriptions

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Create Purchase | | |
| Triggering Event | Purchase of products. | | |
| Brief Description | Allows the Owner to record a new purchase. | | |
| Actors | Owner | | |
| Related Use Cases |  | | |
| Preconditions | Owner has opened the Main Menu. | | |
| Post Conditions | Purchase is saved to the database and now can be queried. | | |
| Flow of activities | Actor | | System |
|  | 1. | Requests to add a new purchase | Displays a list of suppliers and prompts for selection. Prompts for purchase date and selection of receipt file location. |
|  | 2. | Selects a supplier.  Enters purchase date and receipt file selected. | Verifies that a supplier was selected Verifies that date was entered and receipt selected.  Creates a unique identifier for the purchase. Displays the purchase Prompts to enter purchase details. |
|  | Loop | Chooses new product | Displays a list of products, sorted by product type and prompts for selection. |
|  | 3. | Selects a product | Display a list of locations and prompts for select. Prompts for quantity. |
|  | 4 | Selects location and enters quantity and price | Creates a unique identifier for the detail.  Data is valid  Extended price is calculated (price \* quantity ordered)  Taxes are calculated and displayed  Purchase total is updated with the extended price + taxes  Purchase tax is updated with the tax amount  Displays purchase including totals, date and list of products and locations  Prompts to add another location |
|  |  | Repeats above step until all locations for a product were selected | Display a list of products and prompts for selection |
|  | End | When all products are selected | Prompts to save purchase |
|  | 5. | Chooses to save | Saves the purchase and returns to the main menu |
| Exception Conditions | * Owner chooses to cancel adding the purchase | | |

A diagram of a project

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Query Purchases | | |
| Triggering Event | Owner requires a list of purchases for a date period listing Supplier, Products quantity purchased, price paid and the extended amount (price \* quantity) | | |
| Brief Description | Allows the Owner to retrieve purchases for a specified date range | | |
| Actors | Owner | | |
| Related Use Cases |  | | |
| Preconditions | Owner has opened the Main Menu. | | |
| Post Conditions | purchases are retrieved, totaled and displayed | | |
| Flow of activities | Actor | | System |
|  | 1. | Requests to query purchase (by date) transactions | Displays a calendar |
|  | 2. | Selects date range | Verifies that dates are selected  Retrieves purchases in the specified date range, calculating totals  Prompts to exit |
|  | 3. | Request to cancel | returns to the main menu |
| Exception Conditions |  | | |

A diagram of a computer program

Description automatically generated

**Your tasks:**

1. Create a class diagram to support the above case study and Systems Use Case Specifications
2. Create an object level sequence diagrams, to support the use case descriptions described above.
3. Include in your word compatible file, what Visual Paradigm would output.

**.h File**

**Product.h**

Public class Product {

Int productid;

String productName;

String Ingredients;

String Description;

Double Price;

String Instructions;

ProductType productType;

PurchaseDetail purchaseDetail;

String saleDetail;

}

**ProductType.h**

Public class ProductType{

Int productTypeID;

String productTypeDesc;

}

**PurchaseDetail.h**

Public class PurchaseDetail{

Purchase purchase;

Location location;

Int quantity;

Product product;

String receipt;

purchaseDetailId;

}

**Purchase.h**

Public class Purchase{

Int purchaseId;

Supplier supplier;

Int date;

String receipt;

PurchaseDetail purchaseDetail;

Create();

getPurchase();

}

**Location.h**

Public class Location{

Int locationID;

String locationName;

String address;

}

**Supplier.h**

Public class Supplier{

Int supplierID;

Int phone;

String email;

String supplierName;

}

**UIController.h**

Public class retrieveProductTypes() {

// allows the user to request product types

// passes message to the Domain Controller

// displays list of product types to the user

}

Public class addProduct(name,ingredients,description, price,instructions,productTypeID){

// add a new product

// passes entered data to the domain controller

// displays product

}

Public class save(){

// sends request to domain controller to save

}

Public class selectProductType(productTypeID){

//passes selected id to domain controller

// displays list of products for that product type

}

Public class startPurchase(){

// initiates to add the purchase

// displays the list of suppliers

}

Public class selectPurchase(supplierID, date, receipt){

// passes entered data to the domain controller

// displays purchase

}

Public class retrieveProduct(){

// allows the user to request products

// displays list of products

}

Public class selectProduct(productId){

// passes selected data to the domain controller

// displays list of locations

}

Public class enterPurchaseDetails(locationID,quantity,price){

// passes entered data to the domain controller

// displays the totals,taxes,extendedPrice, purchaseDetails

}

Public class save(){

// sends request to domain controller to save

}

Public class retrievePurchases(){

// displays a calendar and prompts for date range selection

}

Public class getPurchases(startDate,endDate){

// passes request for purchases to the domain controller

// displays list of purchases

}

Public class cancel(){

// cancels

}

**DomainController.h**

Public class getProductTypes(){

// requests product types from the entity manger

// retrieves product types data and returns to the UIController

}

Public class createProduct(name,ingredients,description, price,instructions,productTypeID){

// sends request to create a new product

// receives new product

// returns new product

}

Public class generateProductID(){

//creates productid

}

Public class save(){

//sends save to entity manager

}

Public class getProductTypes(){

// requests product types from the entity manger

// retrieves product types data and returns to the UIController

}

Public class getProducts(productTypeID){

// requests products for productype from the entity manger

// retrieves product data and returns to the UIController

}

Public class getSuppliers(){

//requests suppliers from the entity manger

// retrieves suppliers and returns to the UIController

}

Public class createPurchase(supplierID, date, receipt){

//sends request to create a newpurcahse

//receives new purchase and returns to UI

}

Public class getProducts(){

// requests product from the entity manger

// retrieves product data and returns to the UIController

}

Public class getProducts(productId){

// requests products for productype from the entity manger

// retrieves product data and returns to the UIController

}

Public class createPurchaseDetails(locationID,quantity,price){

// sends request to create a new purcahsedetail

//receives new purchase details including extended price and totals and returns to UI

}

Public class generatePurchaseDetailID(){

//calculates new purchaseDetailID

}

Public class extendedPrice=calculateExtPrice(price\*quantity){

//calculates extendedprice

}

Public class taxes=calculateTaxes(){

//calculates taxes

}

Public class totals=calculateTotal(){

//calculates totals

}

Public class updateTotals(){

//updates totals

}

Public class updateTax(){

//updates tax

}

Public class save(){

//sends save to entity manager

}

Public class getPurchases(startDate,endDate){

// requests purchases from the entity manager for the date range

//returns purchases to the uicontroller

}

Public class totals=calculateTotal(){

//calculates totals

}