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.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | | |

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 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  | | |

**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.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A diagram of a project

Description automatically generated

A blue and white diagram

Description automatically generated

A diagram of a diagram

Description automatically generated

A diagram of a diagram

Description automatically generated

#ifndef UICONTROLLER\_H

#define UICONTROLLER\_H

namespace Controllers {

class UIController {

public:

void queryProductTypes();

void addProduct(int newProdName, int newProdDesc, int newProdIngred, int newProdInstruct, int newProdType = prodTypeID);

void save(int newProd);

void querySuppliers();

void addPurchase(int supplierID, int purchaseDate, int purchaseReceipt);

void queryProducts();

void chooseProduct(int prodID);

void addDetails(int locationID, int purchaseDetailPrice, int purchaseDetailQty);

};

}

#endif

#ifndef SUPPLIER\_H

#define SUPPLIER\_H

#include<iostream>

namespace Holiday\_Store {

class Supplier {

private:

int supplierID;

std::string supplierDesc;

public:

void getSuppliers();

void selectPurchaseSupplier(int supplierID);

void getSupplier();

};

}

#endif

#ifndef PURCHASEDETAIL\_H

#define PURCHASEDETAIL\_H

namespace Holiday\_Store {

class PurchaseDetail {

private:

int purchaseDetailID;

Location purchaseDetailLocation;

int purchaseDetailQty;

double purchaseDetailPrice;

public:

void selectPurchaseDetailLocation(int locationID);

void create(int purchaseDetailID, int purchaseDetailLocation, int purchaseDetailQty, int purchaseDetailPrice);

};

}

#endif

#ifndef PURCHASE\_H

#define PURCHASE\_H

#include <iostream>

namespace Holiday\_Store {

class Purchase {

private:

int purchaseID;

int purchaseDate;

Holiday\_Store::Product purchaseProduct;

Holiday\_Store::PurchaseDetail purchaseDetailSet;

std::string purchaseReceipt;

std::string purchaseSupplier;

double purchaseTotal;

double purchaseTax;

public:

void getPurchase();

};

}

#endif

#ifndef PRODUCTTYPE\_H

#define PRODUCTTYPE\_H

#include <iostream>

namespace Holiday\_Store {

class ProductType {

private:

int prodTypeID;

std::string prodTypeDesc;

public:

void getProductTypes();

void getProductTypeID(int newProdTypeID);

void selectProductType(int prodTypeID);

void getProductType();

void generateProdID();

};

}

#endif

#ifndef PRODUCTINVENTORY\_H

#define PRODUCTINVENTORY\_H

namespace Holiday\_Store {

class ProductInventory {

private:

int prodInvenID;

Holiday\_Store::Product productStock;

int prodQty;

int prodPrice;

};

}

#endif

#ifndef PRODUCT\_H

#define PRODUCT\_H

#include <iostream>

namespace Holiday\_Store {

class Product {

private:

int prodID;

Holiday\_Store::ProductType prodType;

std::string prodStatus;

std::string prodtName;

std::string prodInstruct;

std::string prodDesc;

std::string prodIngred;

public:

void createProduct(int newProdID, int newProdName, int newProdDesc, int newProdIngred, int newProdInstruct, int newProdType, int prodStatus = "Active");

void generateProdID();

void getProduct();

void getProducts();

void selectProduct(int prodID);

void getProduct(int prodID);

};

}

#endif

#ifndef LOCATION\_H

#define LOCATION\_H

#include <iostream>

namespace Holiday\_Store {

class Location {

private:

int locationID;

std::string locationDesc;

Holiday\_Store::ProductInventory prodInvenSet;

public:

void getLocations();

};

}

#endif

#ifndef DOMAINCONTROLLER\_H

#define DOMAINCONTROLLER\_H

namespace Controllers {

class DomainController {

public:

void queryProductTypes();

void addProduct(int newProdName, int newProdDesc, int newProdIngred, int newProdInstruct, int newProdType = prodTypeID);

void generateProdID();

void get(int ProductType);

void save(int newProd);

void querySuppliers();

void addPurchase(int supplierID, int purchaseDate, int purchaseReceipt);

void queryProducts();

void chooseProduct(int prodID);

void addDetails(int locationID, int purchaseDetailPrice, int purchaseDetailQty);

void generatePurchaseDetailID();

void calculateExtendedPrice();

void calculateTaxes();

void calculateTotal();

};

}

#endif