The attached code was entirely my own work.

Name: Charles Allensworth

Organization: Revature

Date: 10-04-2019

**project 0: store application**

Presentation:

Introduction, How it was written, tools used, nature of application.  
   
 Hosting Service,   
  
 Demo product's features  
.   
 Conslude with Possible Expansion.

Presentation via Google Hangouts.

**I. Understand the Problem** (Objectives, Specified Tasks, Implied Tasks)

**Problem Specification:**

Create a store kiosk for customers an/or staff to log into in order store and process orders.

Functional Requirements

• place orders to store locations for customers DONE

• add a new customer DONE

• search customers by name DONE (by phone, OKed by Instructor)

• display details of an order DONE

• display all order history of a store location DONE

• display all order history of a customer DONE

• input validation NEEDS FURTHER ANALYSIS <<<< UNIT TESTING?

• must include exception handling (try catch) NEEDS FURTHER WORK

• persistent data (SQL); no products, prices, customers, etc. hardcoded in C# DONE

• logging DONE  
 Log into database, get data from database, place orders on database, get   
 data from database.  
  
 Query, add, modify.

Serialize location and customer information in case of device-reboot

Synch with server.

• (optional: order history can be sorted by earliest, latest, cheapest, most expensive)

• (optional: get a suggested order for a customer based on his order history)

• (optional: save some or all data to disk in JSON format)

• (optional: load some or all data from disk)

• (optional: display some statistics based on order history)

Structural Requirements

**business logic**

• class library DONE

• contains all business logic DONE

• contains domain classes (customer, order, store, product, etc.) DONE

• documentation with <summary> XML comments (optional: <params> and   
 <return>) TODO

• has no dependency on UI or any input/output considerations MET  
  
**data access**

• class library DONE

• contains scaffolded EF DbContext DONE

• contains data access logic but no business logic DONE

• use repository pattern for separation of concerns DONE

**test**

• at least 10 test methods <---

• use TDD for some of the application

• focus on unit testing business logic; testing the console app is very low priority

**user interface DONE**

• interactive console application DONE

• has only display- and input-related code DONE

• low-priority component, will be replaced when we move to project 1

**object model**

**DONE**

customer

• has first name, last name, etc.DONE

• (optional: has a default store location to order from) DONE

order

• has a store location DONE

• has a customer DONE

• has an order time (when the order was placed) DONE

• can contain multiple product types in the same order DONE

• rejects orders with unreasonably high product quantities DONE

• must have some additional business rules

User must log in DONE

Manager must log in DONE

location

• has an inventory DONE

• inventory decreases when orders are accepted DONE

• rejects orders that cannot be fulfilled with remaining inventory DONE

• (optional: more than one inventory item decrements for a given product order, for at least one product)

product

• Product can literally be anything. A generic "product" class will be the first step.

Chosing a specific product is secondary, and can be implemented later, but must   
 be chosen soon.

Instructor mentioned pizzas or food or other services, which are going to have   
 their own related details.

Product is robot parts.

**Deliverable**:

Store kiosk program in C# program and SQL database to store related data for reular query/storage.

**Objectives/ Requirements**:

Core back-end functionality for a point-of-sale kiosk.

**Outputs**:

Data to the database.

Order Confirmation

Changes to Location Inventory.

**Inputs** :

New Customer Information

Customer Orders

Location Information

**Assumptions**:

The kiosk should be as generic as possible, and the GUI is non-critical.

**Constraints**:

Must use C#, core code project must be completed within 7 days. Testing will follow.

**Equations/Data/Previous Results**:

No equations given, except that inventory will reduce, cash-on hand

**II. Design a Solution:**  (**Flow Diagram**, Class or UML, and/or **Pseudocode**)

**Flow Chart:**

Customer Enters Kiosk <--1

|

V

New Customer (2) Existing Customer (1)

| |

V |

Enter Customer Information |

| |

V |

Login Customer <---------------

|

V

Customer Menu------------------------> Get Customer Info   
 | |------> Order History

V |

Place Order ----------- |------> Prompt Customer for Order

| |

V V

Process order Adjust This Location's Inventory

START ----> Menu ----> ( 1 ), ( 2 )

-----> ( 3 )

-----> ( 4 ) ---------> Exit

( 3 )

|

|

V

Existing Location? Yes -----------------

~~No~~ Yes |

| |

V |

Enter Location & Mgr Information |

| |

V |

Login Manger <----------------

|

V

Mgr Menu--------------------->  
 |

| -- Get Location Info

|--> Adjust Location Info

|--> Get Location Inventory

----->Get Order History

**UML Diagram**:

**Test Plan**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Case | Purpose | Test Data | Expected Result | Actual Result | Pass/Fail | Comments |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**III. Implement Solution** (Screenshots and/or Source Code)

Screen Shot?

Sample Source Code?

**IV. Execute Test Plan** (Filled Out Test Plan, as Planned in Part II)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Case | Purpose | Test Data | Expected Result | Actual Result | Pass/Fail | Comments |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |