Kortni Dees, kdd195; Github: kdd195
Samuel Coley, sdc285; Github: DavidColey
Thomas Marandi, tm1383; Github: waveworms
Joseph Sommer, jjs451; Gtihub: jjs451

Assignment 3

Setup Instructions:

Operating Systems used for development: Windows 10, Antergos Linux, and Ubuntu Linux
Programming language used for development: C++
Database/storage method: Static file

To setup the program to run easily, a makefile has been included. So, to compile everything, simply run “make” in the command line. The executable is named “driver” and should be run in the fashion of “./driver username password” with the username and password being replaced with valid credentials. Text files of “users.txt” and “inventory.txt” include initial values for the user and inventory functionalities and should be included in the same folder as the rest of the files. While there are initial values for the cart and order history functionalities uploaded to the Github repo, those are not crucial to running the program, as the program will create new files if those don’t exist.

Lessons Learned:

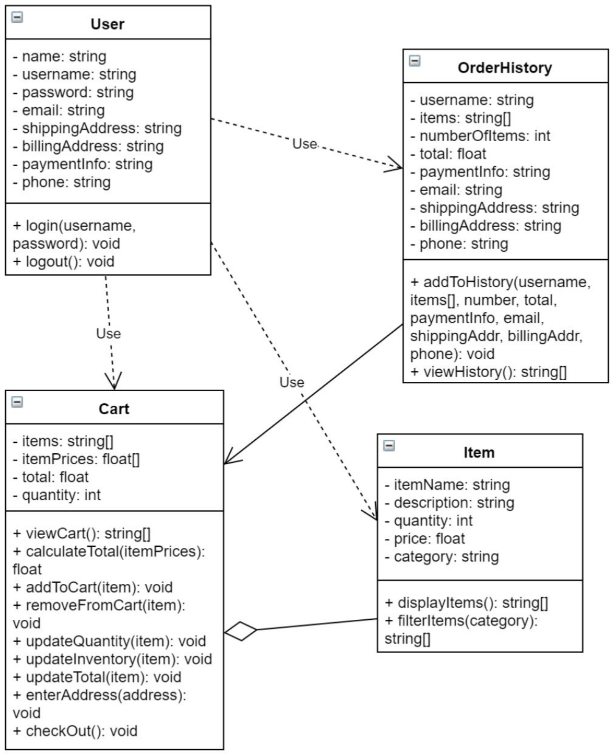
This assignment provided a good look into the design to development phases. Additionally, it provided some real look into requirements changing midway through a project. As far as comparison to other academic assignments, it was more on-par with a class that has a long term project (such as Intro to Software Engineering or DCSP) only on a much faster timeline. This honestly provided a bit more structure to it, as a lengthy timeline can sometimes allow for slacking or losing direction midway through the project.

Some of the benefits of upfront design is you go into the project with a sense of direction. The design done upfront definitely eliminates any confusion on where to begin coding. Additionally, upfront design really does help divide up responsibilities because classes were already planned out. So, basically, the project became a divvying up of the classes and making the individual components work together. Personally, I think that the design process before beginning a project should ease some of these concerns. If it doesn’t, maybe the design or requirements aren’t quite as clear as they should be.

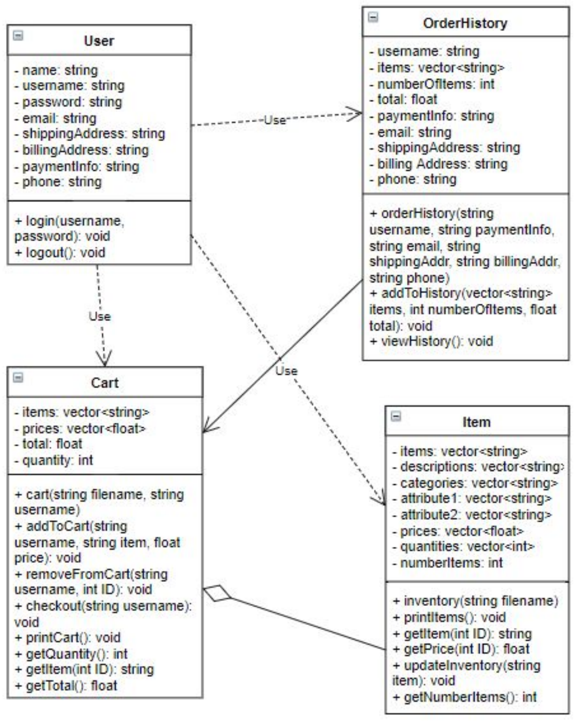
Finally, the group as a whole was fairly familiarly with Github. There was some re-learning process that applied as it had been a time since some of us had dealt with Github. Other than that, there wasn’t much learning that had to go into utilizing Github.

Compare/Contrast Prescribed Architecture:

Assignment 2 UML Class Diagram:



Assignment 3 UML Class Diagram:


User
- name: string
- username: string
- password: string
- email: string
- shippingAddress: string
- billingAddress: string
- paymentInfo: string
- phone: string
+ login(username, password): void
+ logout(): void
OrderHistory
- username: string
- items: vector<string>
- numberOfItems: int
- total: float
- paymentInfo: string
- email: string
- shippingAddress: string
- billing Address: string
- phone: string
+ orderHistory(string username, string paymentInfo, string email, string shippingAddr, string billingAddr, string phone)
+ addToHistory(vector<string> items, int numberOfItems, float total): void
+ viewHistory(): void
Use
Use
Use
Cart
- items: vector<string>
- prices: vector<float>
- total: float
- quantity: int
+ cart(string filename, string username)
+ addToCart(string username, string item, float price): void
+ removeFromCart(string username, int ID): void
+ checkout(string username): void
+ printCart(): void
+ getQuantity(): int
+ getItem(int ID): string
+ getTotal(): float
Item
- items: vector<string>
- descriptions: vector<string>
- categories: vector<string>
- attribute1: vector<string>
- attribute2: vector<string>
- prices: vector<float>
- quantities: vector<int>
- numberItems: int
+ inventory(string filename)
+ printItems(): void
+ getItem(int ID): string
+ getPrice(int ID): float
+ updateInventory(string item): void
+ getNumberItems(): int

Some of the similarities between the diagrams include keeping the user class the same and a lot of the variables and functions the same between classes. Some of the changes just included changing a function from the cart class to the item class (as that ultimately made more sense, implementation-wise) and changing some of the types of the variables. Additionally, some more functions were added to the classes to account to implementation.

The implementation was different because some of the transferring between classes wasn't really taken into consideration. For example, the order history class had to use variables initialized in the cart class. However, the original diagram didn't account for any "getter" function for those variables to obtain that value to use in the order history class. So, those functions were added. Also, some variables changed type in the actual implementation. This was because we accounted for strings in the design, but strings in C++ don't allow them to be dynamically created easily for our purposes. So, ultimately, any arrays were changed to vectors to allow items to be added to those as needed.