Erik Kanamori - ekana003
Grace Lee - glee139

Online cafe application implementation
UserTypes: Manager, Employee, Customer

Print_Menu: This function allows us to print out the current menu with the item name, time price, etc. This way the user will be able to look at what is currently on the menu at any time. This function also prevents the user from being able to edit any part of the menu. This function is included in all three user types. **Erik**

searchItemName: This function allows the user to input an item name. It then returns the details of the item in the current menu, if it can be found. This function only allows the user to search an item by name and not any other attribute of an item. It is not case sensitive. It is also included in all three user types. **Grace**

searchItemCategory: This function allows a user to input a valid type of an item (i.e Drinks). It then returns a table of all the items on the menu that falls under said category. This function only allows the user to search specifically in the section of item types and not any other attribute of an item. It is also included in all three user types. **Grace**

Update Profile: The manager is able to modify any attributes of a selected user such as the login, phone number, password, and favorite items. The employee and customer are able to update any aspect of only their own profile. **Grace**

PlaceOrder: The function allows a user to input a food item to place an order. The user is able to order more than one item at a time. This function is included in all three user types. **Erik**

ModifyMenu: This function allows only the manager to choose whether to add, delete, or modify an existing item. If they wish to add an item then they must input all the necessary information for the new item before it is added to the current menu. If they wish to delete or modify an existing item they must input the item name so that the program can check if the item is currently in the menu. If the item is found, it is either deleted or a prompt will show up asking what aspect of the item they want to change and what to change it to. **Grace**

BrowseOrderHistory: The manager and the employee are able to view all unpaid orders from the last 24 hours. The customer is able to view the 5 most recent orders that they made. The users are not able to modify these orders. **Erik and Grace**

UpdateOrder: The manager and employees are able to change the status of an order from unpaid to paid by inputting the orderid of said order. They are only allowed to change the paid attribute from an order and nothing else. Employees are able to modify a previously made order that has not been paid yet. The function does not allow the user to modify an already paid order

or an order that is not theirs. The user should input the orderid to first choose an order to modify, if valid the user then inputs the modified order and the order is thus changed. **Erik**

Problems/Findings: Some of the problems that we ended up hitting was how to acquire the timestamp from through a SQL Query. We also had trouble figuring out how to compare it to the number given in the csv file. We later discovered that we could keep the current timestamp as a string to make it easier to compare. We also ran into an issue involving our switch cases when trying to compare the usertype a Manager, Employee, or Customer to determine which options to provide them. We discovered that the issue was because cases do not work with a string literal so we had to change our switch statements and cases into if else statements. Another problem that we ran to was inserting and updating queries into Order and ItemStatus. The error we ran into was about foreign keys and we figured out that while checking for lower cases, the user input was changing and therefore not equal to the item in the Menu. We fixed it by grabbing the item from the List that is returned to us and using that as the input for the query. We also ran into the issue about the whitespace after the userType, Manager.