Name: Abdifatah Ahmed

Student ID: 500838495

Section Number: 04

1. **Describing use case diagram(s)**

I have two separate use cases, but they could be forged into one huge use case diagram. I have also taken consideration of the extra functions added in my class diagram that weren’t on my use case diagram. One use case for the Manager actor and the other is for the Customer. Once the Customer logs into his/her account, they can do a wide variety of things such as withdrawing or depositing cash, online shopping and even checking their current balance. The Manager is required to login like the customer to access the bank’s functions. The Manager can add or delete Customers from the banks database. For deleting a Customer, the Manager must search for an already existing customer using their name or account ID.

|  |  |
| --- | --- |
| Use case name | BANKACCOUNT |
| Participating Actors | Initiated by Manager  Communicates with Customer |
| Flow of events | 1. The Manager must sign onto his/her account using the username and password, “admin” and “admin”. 2. The Manager then creates the Customer by giving them a unique UserID, username and password. This newly created Customer has $100 in their account. 3. The Customer can now logon to their new account using their assigned username and password. 4. The Customer can deposit, withdraw and make online purchases. |
| Entry condition | * Manager must be logged in * Customer needs account created by Manager to Login |
| Exit conditions | * Customer logs out with the Logout button * Manager logs out all Customers |
| Quality requirements | * Customer purchases must be greater than 50 plus added fee and must be less than their current balance. * Manager can only Login with “admin” as username and password * Customer can only login if their account was created by a Manager first |

1. **Describing the class diagram**

My class diagram consists of all the classes used in my code. In these classes, I have listed the names of variables and methods with their respective types. Relationships among objects and classes are displayed. The class that I choose to address is the Account class. Examples of relationships are how class Goldlvl, Platinumlvl, Silverlvl are subclasses of the parent class Account. AccountLevel has an aggregate relationship with Account and the multiplicity between the classes are 1 to 1. Class Customer depends on the Account class and shares a 1 to 1 multiplicity with it too.

1. **UML State design implementation**

My UML class diagram implements the State design pattern structure. The Account class is the “Context” with the request method of getLevel(). The AccountLevel class is the abstract “State” which handles the request with the checklevel() method and is an aggregate of Account. Silverlvl, Goldlvl and Platinumlvl are all subclasses of AccountLevel which also handle the request of the Account class by having the checklevel() method.