ASSIGNMENT REPORT

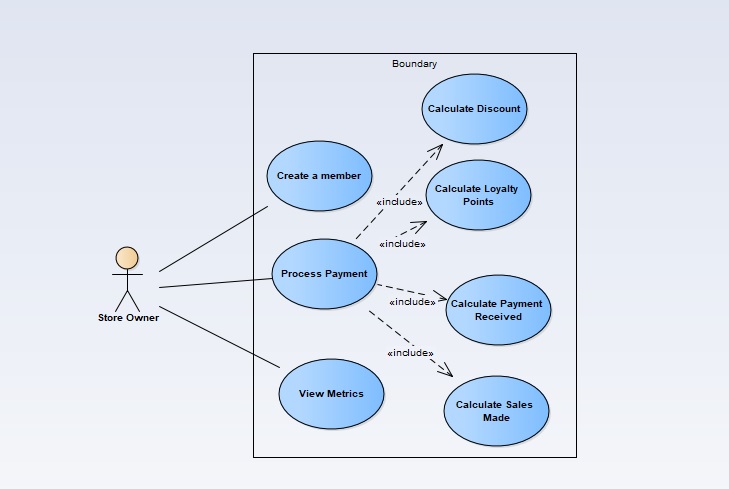
**Use Cases:**

* A menu interface with 4 options like create a member, make a purchase, show metrics, and exit the system.
* The system should allow four type of members to the system like guest, standard member, gold member, and internal employee.
* The metrics to be computed on a purchase:

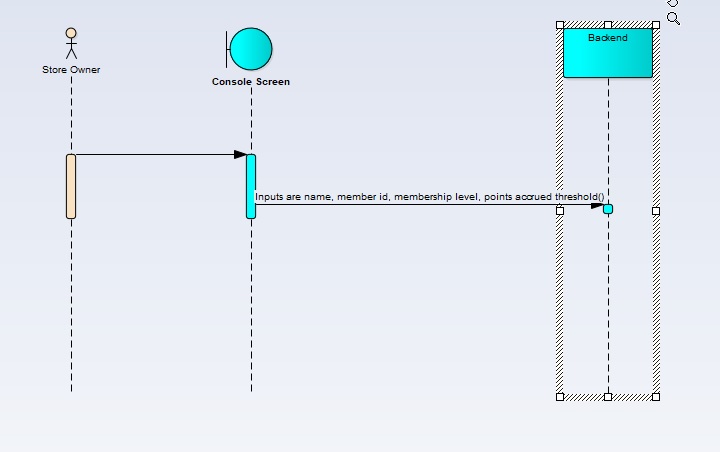
1. Total Sales made
2. Total payment received
3. Total sales made in individual membership tiers
4. Total loyalty points accrued in individual membership tiers
5. Total discounts given in individual membership tiers

**UML Diagrams:**

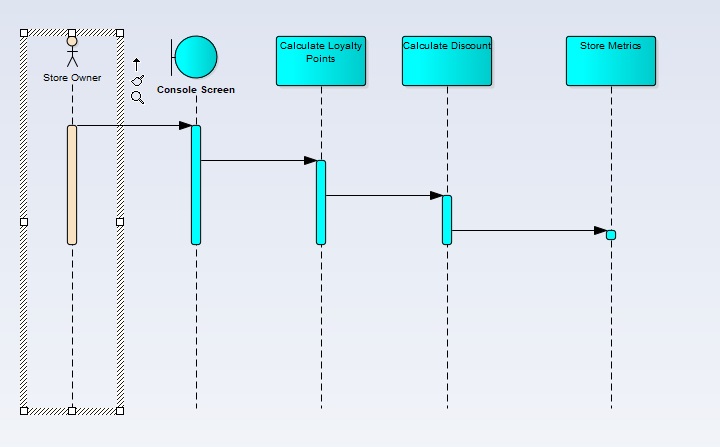
1. **Use Case Diagram:**

****

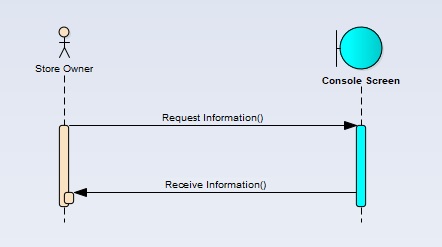
1. **Sequence Diagram:**
2. Create a member

****

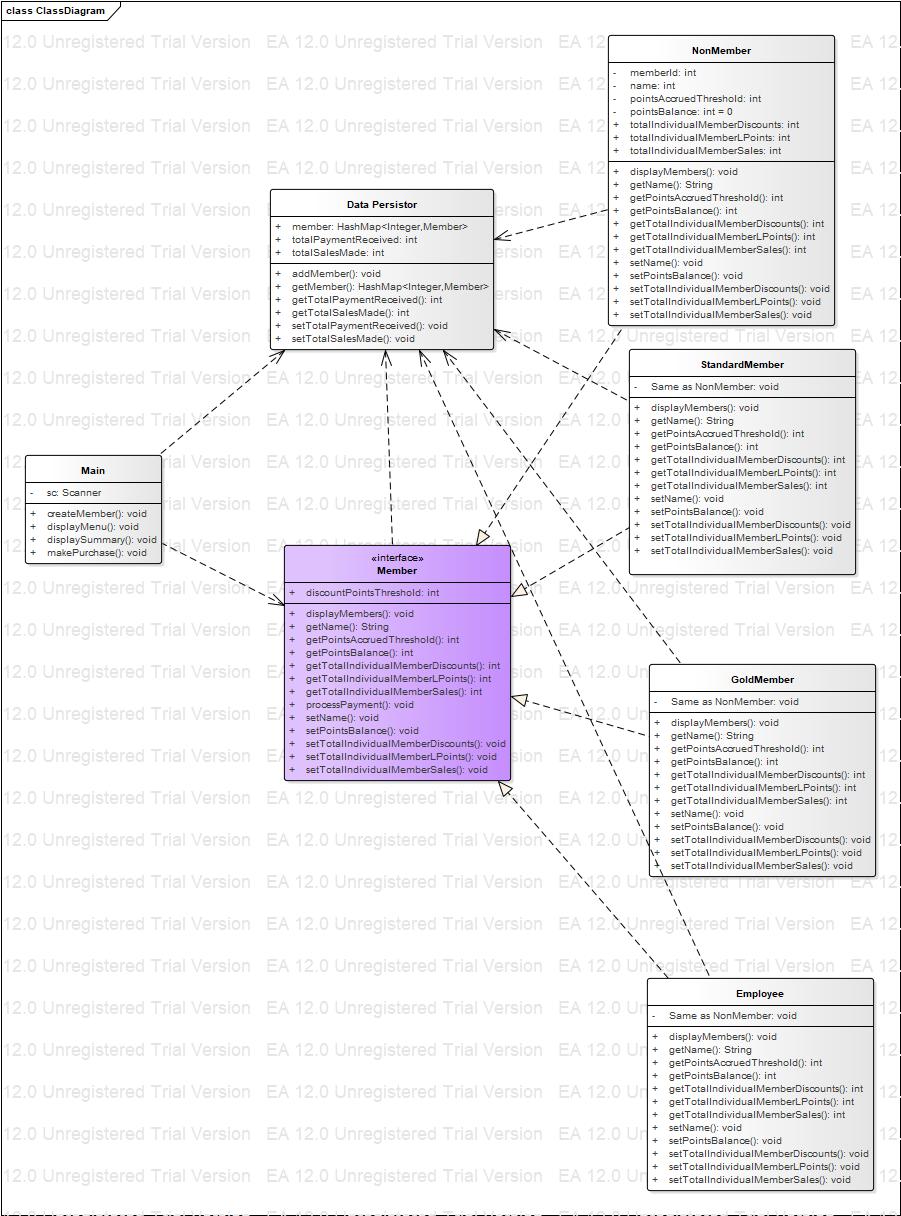
1. Process Payment



1. Display Metrics



1. **Class Diagram:**

****

**Reflection**:

I realised that it is very important to make a design and analysis of requirements as it gives a fair idea and understanding of things involved in programming. UML diagrams is an effective way to do design and analysis. The various diagrams like Class diagram, Use case diagram, Sequence diagram, etc serve this purpose. I got a fair understanding of Interface and Polymorphism in Object Oriented Programming. I have an interface namely Member and the sub-classes are the types of member like Non-Member/Guest, Gold, Standard, or an Employee.

I worked with Polymorphism concept whereby overriding was carried out. All the sub-classes of Member interface overrides the methods of Member interface. In the interface, a static method namely processPayment was used with a definition to calculate loyalty points before and after discounts, purchase value after discount, total discounts and these attributes’ values in each membership tier. This is used by all the membership classes. A class namely DataPersistor was used which is accountable for total sales made and payment received in all membership levels.

I have a fair understanding of Interface and Polymorphism by implementing them in this assignment. UML diagrams helped me a lot in the entire coding process. One significant change I did in my UML diagram was that initially I had individual membership metrics in my DataPersistor class but I realised that the type of membership is only known at runtime and hence I stored them at membership classes level itself. Precisely, UML diagrams ease the programming experience.