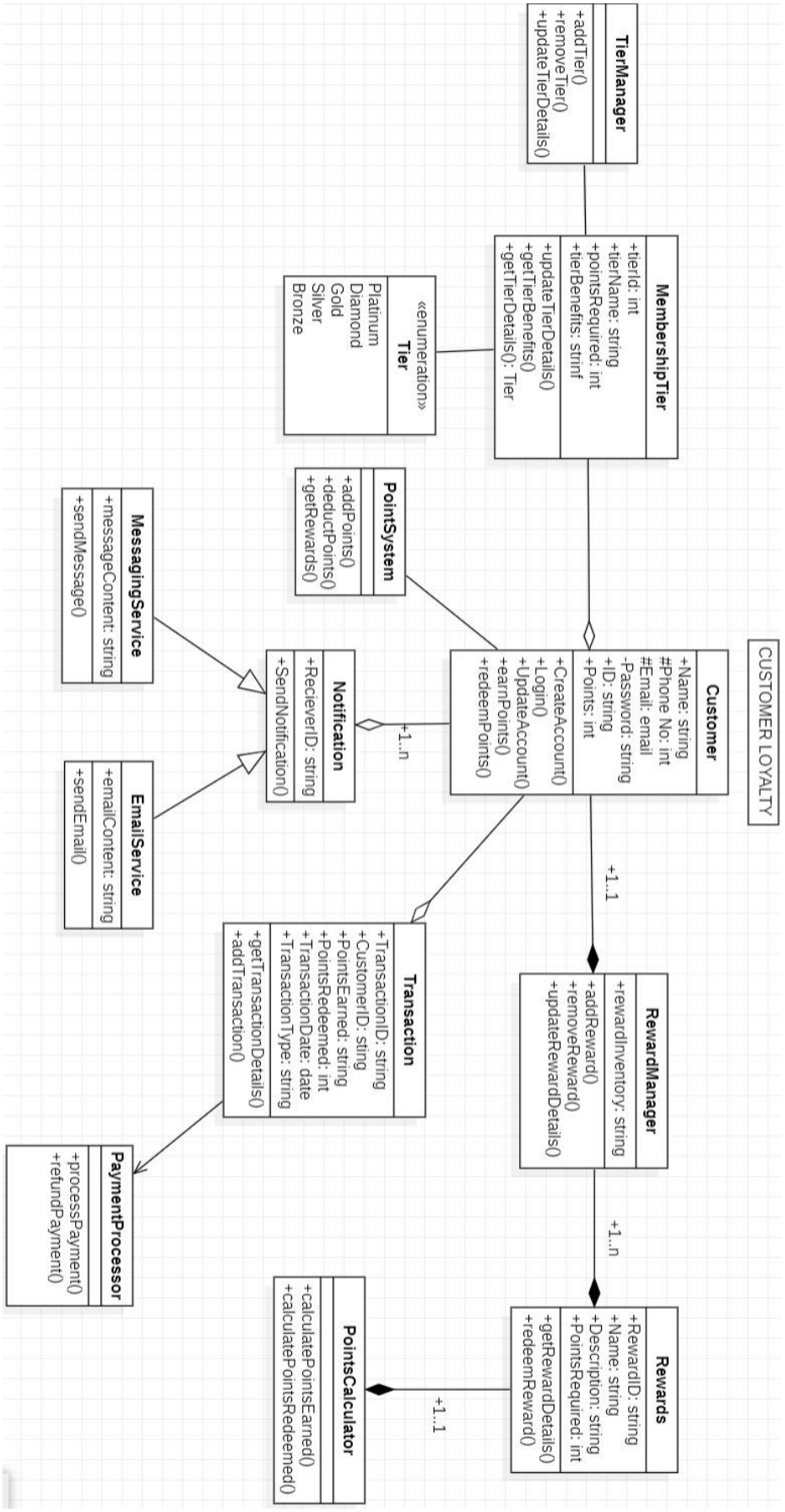
UE20CS352- Project Class Diagram
Object Oriented Analysis & Design

Team Number: *18*
Team Name: *4 Amigos of OOAD*
Topic: *Customer Loyalty Software*

Pratham Bhat	**PES1UG20CS305**
Pratham Rao	**PES1UG20CS306**
Rahul Mallya	**PES1UG20CS318**
Rahul Ramesh	**PES1UG20CS319**

Use Case Diagram:



Model View Controller Format:

Model-View-Controller (MVC) is a software architectural pattern used for designing and implementing user interfaces in a structured way. It separates an application into three interconnected components, namely Model, View, and Controller, each with a specific responsibility.

Model:

The attributes in all of the classes mentioned refer to some data instances that are to be called from the ***Model*** of the software.

Customer	*Membership Tier*
Point System	*Reward*
Transaction	*Message*

View:

The View represents the user interface of an application. It presents the data from the Model to the user in a visual format. The View is responsible for displaying information, receiving user input, and handling events.

TransactionView	*RewardView*

Controller:

The Controller acts as an intermediary between the Model and View components. It handles user input, interprets the commands, and updates the Model and View accordingly. The Controller is responsible for processing user requests, coordinating the flow of data, and managing the overall behavior of an application.

Reward Manager	*Transaction Manager*
Tier Manager	*Payment Processor*
Membership Controller	*Email Service*

INCORPORATING SOLID AND GRASP PRINCIPLES:

Customer Class:

- Follows the Single Responsibility Principle (SRP) by focusing only on customer-related functions.
- Follows the Information Expert principle by being the most knowledgeable about a customer's points and rewards.

Rewards Class:

- Follows the SRP by focusing only on reward-related functions.
- Follows the Information Expert principle by being the most knowledgeable about a reward's details and redemption.

PointsSystem Class:

- Follows the Creator principle by being responsible for creating instances of rewards.

RewardManager Class:

- Follows the SRP by focusing only on reward management functions.
- Follows the Controller principle by being responsible for controlling access to rewards.

Notification Class:

- Follows Open Closed Principle (OCP) as the functions of Notification class is open to extensibility and closed for modifications.
- Follows the Low Coupling principle by being independent of other classes.

Transaction Class:

- Follows the SRP by focusing only on transaction-related functions.
- Follows the Creator principle by being responsible for creating instances of transactions.

PointsCalculator Class:

- Follows the Low Coupling principle by being independent of other classes.