Customer Loyalty Marketplace Application Project Description
CSC 540 - Database Application Programming
(Team Project)

Initial plan deadline: Oct 3
Final Due Date: Oct 31

Next steps:

1. During the next two weeks, begin a team review and discussion of the description and start developing your initial data model designs. It is this process that will help you identify any questions or issues of clarity that we need to resolve in the description.
    a. If you still are not part of a team, then let the TAs know to help match you up with a team. **Team size is 4** (one team of size 5 is allowed but need to contact instructor first for permission).
2. An initial project plan which includes the names and unityids of your team members, initial ER model (will no doubt be updated in the future. Also need not be complete but must be substantially so). One to two paragraphs describing your project plan - remaining tasks as you see them currently and planned task assignment for team members. **Deadline for this is Oct 3rd**.
3. Use the online project forum to discuss issues and questions. However, before asking a question, check the forum to see if the question has already been asked and answered. **After October 3rd, we will assume you no longer have major questions concerning the description.**
4. An application flow document shortly that unifies the overall behavior of application. Sample data to be used for demo will be provided later (approx. one week before the final due date).
5. Submission instructions are provided on the last page
6. This is necessarily a live document and during this time of description review will be updated as issues are identified. It will become more final at the end of your initial plan exercise.

Our application will be to implement a marketplace that manages customer loyalty programs for different brands. Generally, loyalty programs define what activities customers may perform to receive specific types of rewards and how to redeem their rewards. The nature of the loyalty program will differ from brand to brand and customers may participate in loyalty programs of multiple brands concurrently. The marketplace offers building blocks for creating and managing loyalty programs such as loyalty program types, activity types, reward types, reward rule types etc. Then brands can use these building blocks to set up and configure a loyalty program for their brand and outsource its management to the marketplace platform. On the other hand, the marketplace offers a single point for customers to manage the different reward programs they participate in. Consequently, the marketplace services two (non-overlapping) types of users: brands and customers.

Besides these two types of user accounts, there are marketplace admin accounts which the employees of the marketplace use to manage the platform. Some of the key marketplace admin functions include adding a new reward type, a new activity type, adding a new brand and adding a new customer.

Loyalty Programs. Brands can set up one of two kinds of loyalty programs: a *regular* or *tiered* loyalty program. A regular loyalty program has three key components: *loyalty points*, *activity types*, *reward types*, *reward earning* (*RE*) *rules* and *reward redeeming* (*RR*) *rules*. A tiered program includes the features of a regular loyalty program but in addition introduces the concepts of *tiers* (three max) which delineate the level of benefits customers get at each tier. There is a precedence ordering between tiers which implies an inclusion relationship. For example, a tiered program may have three tiers and the following ordering Silver << Gold << Platinum which implies that the Gold tier includes benefits of the Silver tier and Platinum includes benefits of both the Gold and Silver.
Note that the specifics of each brand's loyalty program is selected by the brand. What the marketplace offers are categories of artifacts that can be used as building blocks for customizing brand loyalty programs. For example, one brand may set up a tiered program but with only two tiers and give them names *alpha* and *beta* while another may set up a three-level tiered program with names *bronze*, *silver* and *gold*.
The function of the marketplace is to provide the interface for all brands to set up and manage their programs and for customers to interact with programs.

Brands may select a subset from the list of available activity categories to add to their programs: *purchase*, *leave a review*, *refer a friend*. While these three represent the current scope of available types, we must allow for the possibility that the marketplace operator wants to add more types in the future. Similarly, a subset of reward categories can be selected from the list of possible categories: *gift card*, *free product* (new types may also be added later). Both activity categories and reward categories have globally unique codes *(alphanumeric)* and a name.
The marketplace offers customers the opportunities to perform activities and tracks relevant information. For each customer activity, an automatically generated unique identifier, customer id, date, and activity category code is stored. There is also the data corresponding to the activity itself that is captured but varies based on activity category. For example, if it is a purchase, it is an amount i.e. float. If the activity is a review, then it is a string which is the review content, while if it is a "refer a friend", then it is a customer id (of the customer referred).

As mentioned earlier, there are different categories of rewards. A brand can select a subset of reward categories to be used in their loyalty programs and create as many reward instances of their chosen categories as they wish. For example, a brand that chooses to use gift cards as part of their rewards will instantiate a certain number of such gift cards to be available for use as rewards. For each reward instance, its reward category code, the instance's brand-unique identifier, value (amount for gift cards, product name for free product) are recorded. Gift cards also have an expiry date.

RE rules allow a brand to indicate their desired mapping from activity type to reward type. RE rules have three key components: a brand unique reward rule code (alphanumeric string of up to 6 characters), number of points, activity category (essentially stating the mapping between activities and points). This information indicates what points should be automatically added to the wallet of a customer when they perform the corresponding brand activity.
Similarly, RR rules also have three key components: brand unique rule code, points, reward; mapping rewards to points.
Rules (both kinds) also have version numbers (integer(3)). This is because rules can be updated and we would like to keep an account of all versions of rules used. Rule updates modify either the points or rewards or activities in the rules e.g. increase the number of points needed for a reward or change a reward.
In summary, there are four key functions associated with rules (details given in the next sections):

- addRERule(): adds brand new rules
- updateRERule(): allows modification of all fields of a rule (except the code) and increments the version number
- addRRRule(): adds brand new rules
- updateRRRule(): similar to the RER equivalent.

Brands -
Brands are a key user category on the marketplace platform and are given globally unique ids. They each also have a name and an address and join date (date they joined the platform).
A brand sets up its loyalty program on the marketplace (only one per brand allowed) by first indicating the type (tiered or regular) and then adding remaining components to it e.g RR and RE rules. If a tiered loyalty program, a precedence order must be selected to be accepted.
The following functions are available to brands for configuration.

- addLoyaltyProgram:
    - add the type (tier or regular)
    - add RE rules
    - add RR rules
    - add a tier (brand-unique name, level (integers 0,1 or 2), pointsrequired, multiplier): Pointsrequired is the number of points to enter the tier. Multiplier is a points multiplier which multiplies the number of points for a given activity. For example, if an activity earns 10 points at the base tier and a higher tier has a factor of 3, then the activity earns 30 points for a customer at that tier. This feature should only be available for tiered programs.
- validateLoyaltyProgram: because a loyalty program set up is a multistage process, once its instantiation begins, it is assigned a state of *inactive*. After the brand owner feels satisfied that all the components have been entered as desired, an attempt is made to validate the program. For example, if the program was instantiated as a tiered program but no tiers were entered, the program would not be validated or if the tiers do not have distinct, totally ordered levels. Also, there must be at least one RE rule and one RR rule entered.

- addRewards: given reward category, #instances to generate (and value in the case of gift cards), generate a set of #instances of such rewards.

Customers - Customers are identified by a globally unique id, and have a name, phone number, address, and a globally unique wallet id. A customer wallet maintains information about a customer's activities for each brand in whom's loyalty program the customer is enrolled and the points earned so far. Specifically, it includes the activity, loyalty program code, date of activity, points earned for activity, and the RE rule code used for assigning points for that activity. If the program is a tiered program, a tier status is also associated with a customer and when enough points are accumulated to promote a customer from one tier to the next tier, the system automatically updates their tier status. (It is important to think carefully about to structure the design so as to avoid redundancies - as a general principle)
Things customers can do:

- Login to the marketplace -
- Enroll in a brand's loyalty program enroll(brandid, customer): entry in wallet as special activity JOIN and reward points 0.
    - if a tiered program, the customer is automatically assigned to the base tier.
- Reward activities automatically update the customer's wallet with relevant information
    - Purchase activity for a brand: newpurchase(brandid, amount, date, giftcard code)
        - Customers may use an earned (but unexpired) giftcard during the purchasing activity. If so, that giftcard should no longer be available after purchase. (do not worry about balance of giftcard - we assume every use of a giftcard is destructive. However, the amount that will count for the customer towards rewards should be amount less giftcard contribution. In other words, giftcards which are already rewards in themselves cannot now be interpreted as amounts to be used to get future awards.
    - Refer a friend: refer(newcustomerid, loyaltyid, date)
    - Leave a review: review(loyaltyid, review, date)
- View wallet: displays the contents of the wallet
- Redeem points: customer redeems their points for rewards with redeem(loyaltyid, rewardid). This uses the active RR rule and checks if the customer has enough points to redeem. If so, the points are deducted and the reward is considered consumed (no longer available).

<u>It is important to note that in the description of entity types and functions, the list of attributes or parameters given do not reflect the exact relation definition as some attributes that relations may need (particularly those needed to link to other relations) are not explicitly mentioned. Your designing process however, should reveal a complete picture of what you need in each relation.</u>

Sample Queries

Queries on your database will help assess the quality of database design. However, it isn't possible to leave that implementation to only demo day. Therefore, you are being asked to

implement the queries given below as part of your project ahead of the demo. Note that on the demo day, you will be given 2 - 3 queries not listed here to implement directly on your database.

1. TBA.