CPS714 Lab 9

Tech Stack

Backend: Django (python)
Frontend: HTML, CSS, JS

Schema Update

We made a couple of changes:
• reward_id (PK) - INT, Auto Increment
• user_id (FK) - INT, References user(user_id)
• points_earned - INT
• points_redeemed - INT
• reward_description – VARCHAR (255)
• is_active - boolean
• earned_date - TIMESTAMP, Nullable * We made this nullable, in case the reward entry is for a redemption not earning
• redeemed_date - TIMESTAMP, Nullable

We also added a new table called RedeemedItem:

- reward_id (FK) - INT, References reward(reward_id)
- amount - DOUBLE
    - This is the amount that the coupon is valid for - Ex. $5.00, $10.00
- item_description - VARCHAR (255)
    - Could be a coupon or donation
- item_code - VARCHAR (16), Nullable
    - For coupons, they have a coupon code to enter when they purchase

NOTE: RedeemedItem to Reward is a one-to-one relationship, so every redeemedItem is associated with a unique reward table entry

- The only time we need to directly access the RedeemedItem entry when we don't know the reward_id beforehand, is when we are using a coupon, but we will know the coupon code (item_code) which is a unique attribute so it acts as the primary key in that case

Development Design

For Point accumulation, we have designed 2 API calls for this purpose:

- rewards-api/earn-points/ --- Expects post with { user_id: INTEGER, points_earned: INTEGER, reward_description: STRING }
    - This is for generic purposes
- rewards-api/purchase/ --- Expects post with { user_id: INTEGER, reward_description: STRING, price: DOUBLE }
    - This is for purchases made, there is a calculation done to calculate the points earned based on the total price of the purchased object(s)

For the rewards catalog, we will have a rewards page where users can redeem points for either discounts/store credit, or donations, we have API calls for this:

- rewards-api/redeem-coupon/ --- Expects post with { user_id: INTEGER, points_redeemed: INTEGER, reward_description: STRING, amount: DOUBLE }
    - For redeeming coupons
- rewards-api/redeem-donation/ --- Expects post with { user_id: INTEGER, points_redeemed: INTEGER, reward_description: STRING, donation_amount: DOUBLE }
    - For redeeming donations

For Redemption Tracking, we will use the rewards page to show the redemption history for the user as well as the status of the coupons redeemed, i.e. whether they were actually used yet (active or inactive). We will also notify the user when a coupon/donation has been redeemed, which will tell the user how many points were redeemed and the coupon code, if applicable

- We have an API call for using coupons which updates their status to inactive:
    - rewards-api/use-coupon/ --- Expects post with { item_code: STRING }
- We also have an API call for seeing redemption history:
    - rewards-api/get-redemptions/ --- Expects post with { user_id: INTEGER }

For Bonus Points For Engagement, we have API calls:

- rewards-api/submit-feedback/ --- Expects post with { user_id: INTEGER, reward_description: STRING }
    - For users submitting feedback
- + for other engagement opportunities, the generic rewards-api/earn-points/ API call can be used, where the component invoking the call will decide the amount of points earned

Integration

For integration, we have discussed with each of the teams that we have to integrate with and have agreed on API calls to deal with the interaction between our subprojects:

For the User profile team, we can set up an api call that accepts a user id and returns the amount of points they have, and redemption history. We have set up this call for that purpose:

- rewards-api/get-user-info/ --- Expects post with { user_id: INTEGER }
    - This returns the total_points of the user as well as the redemption history as a list of reward table entries, but showing these attributes: { 'points_earned', 'points_redeemed', 'earned_date', 'redeemed_date', 'reward_description' }

For the admin dashboard, we will use the same api call, since they'll also be looking at the point redemption history for each user individually

For feedback review team, we made an api call that they will call when a user submits some feedback. This will earn a flat 10 points for each time a user submits feedback (Subject to change)

- rewards-api/submit-feedback/ Expects post with { user_id: INTEGER, reward_description: STRING }