Jaime “Jay” Lara Friday, June 22, 2018

Cognitive Scale QA Assignment (#2)

Instructions:

Assume you are part of an engineering team that is building a loyalty app for a large retailer. You are in a meeting in which the following stories are being discussed by the product owner and engineering team:

a) As a customer, I want to enroll in the loyalty program.
b) As a program participant, I want to check my balance of reward points.
c) As a program participant, I want to redeem some of my points for a reward.

Describe how you might participate in this meeting to ensure that the development work for these stories can be demonstrated to the product owner.

- What are your areas of concern?
- How would you address them?
- Provide your answers as a PDF.

Submission:

How I see the flow of the stories

a) Participant = Customer.Enroll(LoyaltyProgram)
b) RewardPoints = Participant.CheckRewardBalance()
c) Reward = Participant.Redeem(NumberOfPoints)

Thinking about all of the moving parts for this assignment was a bit tricky. Trying to demontrate these pieces to the product owner seemed easiest when split into sub-tasks (since these are stories). Assuming that Acceptance Criteria is not set in stone, I have taken the liberty of making some implementation suggestions along with these sub-tasks. Each portion will also have subtasks for testing API automation, UI automation, and manual testing.

The CUSTOMER becomes a PARTICIPANT in our LoyaltyProgram™

- Concerns:
    - Will the customer be able to enroll at the register?
    - Can the customer enroll online?
    - Security (Enrolling online vs In-store)
- Solutions:
    - Create an API endpoint (/register)
        - Requires OAuth token via Authentication header
        - Simple POST request. JSON Payload consists of name, address, phone, store info, date created, card code, etc. Response returns an echo of the information entered along with the user’s UUID.
    - Create a Frontend
        - Should be created for data input. User can register via website. If at register, a stripped down version can used
    - Create DB tables
        - AUTH table: Depending on complexity, auth table with have all user information.
        - POINTS table: linked to CARDS table via card code

        - CARDS table: linked to AUTH table via user id
    - Testing:
        - API (/register): Data validation and auth tests
        - UI: Registration web page, Login Page
        - Manual: In store User Acceptance testing

LoyaltyProgram™ PARTICIPANT wants to check POINT BALANCE

- Concerns:
    - Will the customer have a card (with code)?
    - Can they use their email address to check their reward points?
    - Security. Should the user be logged in?
- Solutions:
    - Create API endpoints (/users/<id>/points, /cards/<code>/points)
        - Requires OAuth token via Authentication header
        - Simple GET request. No payload, but may use token to find/verify user. Response returns basic user info (id, name, etc) and number of points.
    - Create a Frontend
        - User can get point balance using card code, user id, or email address.
        - User can check balance via website. Using card code doesn't require login.
        - If at register, user can only use card code. Maybe the register can have an email to card code lookup
    - Create tables
        - AUDIT table: linked to CARD table via card code. Records information related to checking point balance.
    - Testing:
        - API (GET /users/<id>/points): auth, reponse tests
        - API (GET /cards/<code>/points): response tests. Verify correct number of points
        - UI: Check Balance web page (card code), Check Balance Page (logged in)
        - Manual: In store User Acceptance testing. Email to card code lookup screen (if implemented). Lookup balance via card code

LoyaltyProgram™ PARTICIPANT wants to redeem POINTS for REWARD

- Concerns:
    - Will customer be able to purchase items in store?
    - What information will the customer need to purchase an item? Email? Card code? Physical card?
    - What's the conversion method? 1 point = $1?
- Solutions:
    - Create an API endpoint (/rewards/purchase)
        - Requires OAuth token via Authentication header
        - Simple POST request. JSON payload with reward id, user id, card code, method (in store, website), etc. Response returns an echo of the information request and remaining points.
    - Create an API endpoint (/users/purchases)
        - Requires OAuth token via Authentication header

    - Simple GET request. No payload, but may use token to find/verify user. Response returns purchases made by user.
- Create a Frontend
    - If at register, user can purchase item using email address (cashier looks up account/card)
    - User can purchase items via website (requires login). User is sent through a checkout process
- Create tables
    - PURCHASES table: linked to CARD table via card id and AUTH table via user id
- Testing:
    - API (POST /rewards/purchase): data validation, auth, reponse tests
    - API (GET /users/purchases): response tests. Verify all purchases made are in response.
    - UI: Reward selection web page, checkout process pages
    - Manual: In store User Acceptance testing. Purchases can be made
- Other:
    - I would assume that conversion method would be 1point = $1