CMPE 275 Section 1 Spring 2020 Term Project - CartShare

Last updated: 04/30/2020 (status: published)

With the pandemic of COVID-19, a routine trip for grocery shopping becomes an adventure with risks of contracting or spreading viruses. Even in regular times, grocery shopping can be time consuming and costly due to the time and gasoline spent in driving, exacerbated by traffic congestion in heavily populated cosmopolitan areas. Very often, a person making a shopping trip has extra cart room and trunk space to take more items. Given that it happens very often that people from the same neighborhood shop at the same stores around roughly the same time, it can significantly increase the efficiency if the neighborhood shoppers do cart pooling by taking turns to pick up the in-store orders for the participating neighbors. Assuming every trip takes care of 4 shoppers' orders in average and one generally goes for shopping about once per week, the shopping trip you need to make gets reduced to roughly once per per month. So besides the timely benefit of fighting against virus spreading to save lives, it also saves time, mondey (gas and time both translate to cost), reduces Carbon dioxide consumption, and increases neighborhood bounding too.

In this project, you will build a shopping pooling service that allows neighbors living close by to take turns in shopping and delivering groceries. To make it easy to understand, it is similar to combining InstaCart and a Waze Carpool, except that we pool groceries, not the shoppers, and the deliverers are the shoppers themselves, who are spreading the responsibilities, opposed to doing delivery for profits.

The service needs to be hosted in the cloud, and you also need to provide a client app (web app, mobile, or both) for shoppers to use. There are no language or technology requirements for UI implementation. The primary language you use for server implementation, however, *must* be Java. You do not have to use Spring, but you need to exercise the principles, patterns, and methodologies that you have learned in the class, including DI, AOP, MVC, ORM, and transactions. You *must* use a relational database, e.g., MySQL, Amazon RDS (supports many popular relational databases), MSFT SQL Server, or Google Cloud SQL.

If any feature described below is unclear or ambiguous, and you fail to get a clear answer from the instructor or TA, you can use your best judgment to interpret it and add the missing details, provided that you clearly document and explain your reasoning in your project report.

The main features include:

1. User Roles and Authentication
    a. There are three user roles, *admin, pooler,* and *pool leader.*

    i. An admin can add and manage grocery catalog and inventories.
    ii. A pooler can create a shopping pool, thus becoming the coordinator of the pool.
    iii. A pooler can also join an existing shopping pool, thus becoming a member of that pool. The request to join needs to come with at least one reference, and approval from the pool leader. The reference can be the pool leader.
    iv. A pooler can be a member of only one shopping pool at any given time.

b. User registration info
    i. A user needs to provide the following info upon registration.
        1. Screen Name - alphanumeric. Must be unique and cannot be changed. Visible to all pool members.
        2. Nickname - text. Editable. Still must be unique within the service. Visible to all CartPool members.
        3. Email address - used for verification and notification purposes. Invisible to other pool members through the service's UI.

c. You must use Oauth/OpenID for user sign-up and authentication; specifically, you need to support signing in with Facebook and Google. In addition, you ***must*** also support the option of letting the user set up a "local" password in CartPool with his email as the user name and his own password.

d. Account verification
    i. No matter which registration option is used, local account with own password, or signing in with Facebook or Google, CartPool must send an email to the user with a verification code or link.
    ii. The user needs to use that verification code or link to verify his account registration.
    iii. Before an account registration is verified, a pooler cannot join or create any pool, and an admin cannot get any duty done either.

e. It is OK to use libraries like Firebase Authentication to assist user auth.

f. For simplicity, users with an email address in the @sjsu.edu domain *automatically* assume the role of an admin, and users from other domains can only be registered as poolers. This also implies a single registered user cannot function as both an admin and a pooler with the same email address; i.e., users who are both admins and poolers need to use different email addresses for different purposes.

g. Messaging and notification
    i. A pooler can directly message any other pooler through the other party's screen name. For simplicity, it is OK to let all actual messaging and notification be transported through email. For example, when Alice messages Bob through CartPool, it ends up sending an email to Bob through the system.

2. Inventory Management
    a. Stores
        i. An amin must be able to create, edit, and delete a store.
        ii. A store has the following attributes
            1. ID - integer and primary key.
            2. Name - string. Must be unique across the service.
            3. Address
                a. Street number and name
                b. City name
                c. State
                d. Zip code.

            Please note that accurate addresses are essential since we need to use them for deliveries.
        iii. Deletion of a store is allowed only when there are no unfulfilled orders for this store.

    b. Product Catalog
        i. An admin must be able to add, edit and delete products.
        ii. A product has the following attributes
            1. StoreID - as defined above.
            2. SKU - Store-keeping unit number. Must be unique within any given store. StoreID and SKU together can serve as the primary key for products.
            3. Name - name of product, e.g., organic full fat milk.
            4. Description - text description
            5. ImageUrl - an image link to show the product.
            6. Brand - text. Optional.
            7. Unit - Piece, Pound, Oz, etc. Piece is also used for all prepackaged items such as bottle, box, basket, etc.
            8. Price - price per unit. Float, double, or any other real number. In US dollars.
            9. Additional attributes - you can add other attributes as you see fit.
        iii. A product cannot be deleted if there are unfulfilled orders for this product.
        iv. To simply product catalog creation, it is required for an admin to create one product for multiple stores at the same time; i.e., create a product with the same SKU, but assign it to multiple *explicitly* specified stores.
        v. An admin must be able to browse all products, and search products by store ID, SKU, and/or name, so that it is convenient for him to locate a product and make changes as needed.
3. Pools and Membership
    a. A pool has the following attributes
        i. Pool ID - an alphanumeric string specified by the pool creator. Must be unique across the service. Pool ID is not editable once created.

        ii. Name - text. Editable. Must be unique too.
        iii. Neighborhood name: text.
        iv. Description: text
        v. Zip code - US zipcode, a 5-digit number. Cannot be changed.
        vi. Additional attributes - add as you see fit.

   b. Creation of a pool
        i. A pooler not associated with any pool can create a new pool and automatically become the pool leader.
        ii. A pool leader can also function as a pooler within a pool.
        iii. A pool leader can delete the pool he leads only when there are no other pool members.

   c. Join a pool
        i. Any registered and verified pooler can browse the pools, search by pool name and or neighborhood name or zipcode.
        ii. When applying to join a pool, a pooler can either provide the screen name of an existing pool member as a reference, or be given the option to check that he/she knows the pool leader, in which case the pool leader is treated as a reference.
            1. The reference pool member will receive an email notification about the applicant pooler's request to join the pool and use him/her as a reference. The reference pooler can either support or reject the request to join.
            2. The pool leader cannot approve a membership request until the reference has supported the request.

4. Shopping and Order Placement
   a. All users can browse and search for products, but only poolers and pool leaders who belong to a pool can add products to cart and check out.
   b. For simplicity, it is OK to restrict an order to products belonging to a single store only; i.e., you can ask the shopper to pick a store so that product browsing, search, and ordering are all restricted to that store.
   c. When adding a product to a cart, quantity needs to be specified unless the unit is a piece, in which case it defaults to one.
   d. When checking out, a shopper must be able to adjust the quantities of each product, adding or reducing pieces, pounds, etc, and to delete products from the cart too.
5. Order Pooling, Cost, and Insurance
   a. For simplicity, we can use a universal tax rate of 9.25%, a convenience fee rate of 0.5%, which gets a total of 9.75% tax and fee. Presumably, we can use the fee to pay for the cost of the service development and operation. :) Ideally, we want to insure the deliveries as well, so that losses related to delivery damages are covered.

   b. Whenever a pooler delivers an order for a fellow pooler, he earns one contribution credit. When a pooler gets his order delivered, he uses a contribution credit. When a pooler's credit goes negative by 4, i.e., <=-4, his contribution status becomes yellow, and when negative by 6, his contribution status becomes red. The contribution credit and status can always be checked through the app.
   c. When finishing up an order placement, the pooler will be given the option to pick up the order by himself or let a fellow pooler pick up and deliver within 2 days. The contribution credit and status will also be shown to the pooler right at this moment. If the pooler's status has turned yellow or worse, and he still decides to let a follow pooler to deliver, he will be prompted with a popup that reminds him of this contribution status and asks him to confirm his choice.
   d. If a pooler decides to pick up his own order, he will be presented with a list of orders for fellow poolers available for pick-up, in the order that were placed. Only product item info for each order (e.g., product name, image, quantity, etc) is shown, not the buyer's info. The pooler can pick up to one, two, three, .., up to ten orders or what's available, following the order they were placed. For example, if the pooler decides to pick up three, they must be the first three available orders.
   e. After these, the pooler will get order confirmation through email. Which contains info of the order details, as well as other orders to pick up, or the fact that the order will be delivered by some fellow pooler. Up to this moment, no info regarding which fellow poolers made the orders to deliver should be shown or sent in the email.
6. Order Pickup and Delivery
   a. When a pooler goes to the store to pick up his order, there should be a shop assistant to help with the order pickup.
      i. For simplicity, we emulate this by requiring the pooler to go to the pick up menu, select his own pickup order (just one at a time if there are more than one), and click the check out button, which will then show a QR code of his order ID that he can show to the assistant to locate his oder, as well as any other associated orders (also shown on the screen) by fellow poolers.
      ii. This completes the pickup, which triggers email notification to the pooler as well as fellow poolers that the orders have been picked up.
      iii. A detailed delivery instruction will then be emailed to the pooler as well, which includes each order's item detail, and buyer's addresses too for delivery.
   b. UI needs to be provided for a pooler to see his delivery tasks.
      i. Each order to deliver needs to contain the pooler's nickname and delivery address.
      ii. After selecting an order, the pooler can mark it as delivered, which triggers an email sent to the recipient pooler.

        c. UI needs to be provided for a pooler to track the status of his owner orders: placed, picked-up by oneself, picked-up, and delivered.
            i. After an item is marked as delivered, the recipient can mark it as delivery-not-received, if he has not actually received the item, in which case an email will be sent to the delivery pooler.
7. Reputation and Pool Integrity

~~Bonus Features~~

Please note this is the end of functional specification.

User Interface and Experience

In addition to the functional features, we want the app to have proper and intuitive UI/UX design, navigates naturally, runs stable and smoothly.

Grouping

This term project is group based, with group size up to four people. Once the project plan is submitted, group membership cannot be changed.

Source Code Management

You are recommended to use a Source Control Management (SCM) system to manage your team's source code. This can be a private Bitbucket repository or your local git. During the grading of the term project, you may be asked to provide commit history or any other document to help evaluate each team member's contribution.

Cheating Policy

Your app must be built by yourself, and cannot be based on the code base of any existing app. If you used any code not written by yourself, it must be clearly documented in your README.TXT file, unless it is part of publicly available libraries. *If your app is already used to serve the requirements of any other class, it will not be accepted by this class*. In the case any form of cheating is confirmed, you will get an F grade for this class.

Deliverables and Grading

The project is worth 31 points in total. The actual *due dates* of the deliverables will be specified in Canvas.

Group Formation (1 point)

Please submit your project name and group formation info through Canvas.

Project Presentation and Demo (5 points)

To be presented in class.

- The presentation should cover introduction, high-level design, and major features with screenshots. Time limit: 3 minutes.
- You must also do a live demo. The guideline for how to do demos is to be announced. Time limit: 3 minutes.
- Grading will be based on the successfulness of the demo, the content and clarity of the slides, and the delivery of the presentation

The presentation slides must be submitted through Canvas as a PDF file.

Project Report (5 points)

The report needs to cover the following topics.

1. Motivation and introduction of your app
2. High level and component level design
3. Technology choices
4. Description of features with final screenshots
5. Testing plan executed and results
6. Lessons learned and possible future work

You are recommended not to exceed 20 pages, but you will not be penalized just because the report is too long or too short, as long as the level of coverage for the required topics is reasonable and clear. The report must be submitted through Canvas as a PDF file.

Project App (20 regular points ~~+ YY bonus points~~)

Note: the instructions for submission is still *subject to change*. Please refer to submission instructions in Canvas.

1. You must submit all your source code / resource files through Canvas
2. Features correctness, stability, performance, choice of technology and implementation are worth 18 points; **all** grading will be done through your UI - **no** partial grades will be given for any feature not exposed or untestable in UI.
3. User experience is worth 2 points

~~4. The bonus features are worth 2 points~~

5. You need to keep your app live for at least a week before we finish grading
6. README.TXT, including
   a. The names, email IDs, and students IDs of the members
   b. The URL to access your app
   c. Any other instructions necessary for the TA to grade the app
   d. Build instructions
   ~~e. Description of bonus features you implemented, if any.~~