Standard Course Project Description: Mini-Amazon

CompSci 516 Spring 2022

See the overall Course Project Description (file name starting with 1) for general information on the course project. This document is specific to the standard project option.

Overview

In this project, you will build a miniature version of Amazon. On this website, sellers can create product listings and an inventory of products for sale. Users can browse and purchase products. Transactions are conducted within the website using virtual currency. Users can review products and sellers who fulfill their orders.

This project is designed to help you divide work among a team of 4. Each of you will be responsible for implementing a subset of the functionalities described further below:

- Users Guru: responsible for Account / Purchases
- Products Guru: responsible for Products / Cart
- Sellers Guru: responsible for Inventory / Order Fulfillment
- Social Guru: responsible for Feedback / Messaging

That being said, you cannot work in full isolation. An excellent score requires both individual efforts and close collaboration, including integration and testing your code together.

Functionalities

Account / Purchases

Basic requirements:

- A new user can register for a new account; an existing user can log in using email and password.
- Each user account has a system-assigned id. Other account information includes email, full name of user, address, and password. Users can update all information except the id. Ensure that email is unique among all users.
- Each account is associated with a balance. It starts out as $0, but can be topped up by the user. The user can also withdraw up to the full balance. In practice these actions would require some real payment mechanism, but it is not required for this project.
- Users can browse their history of purchases, sorted in reverse chronological order by default. For each purchase in this list, show a summary (e.g., total amount, number of

items, and fulfillment status) and link to the detailed order page (see Inventory / Order Fulfillment).
- Users can also search and filter their purchase history by item, by seller, by date, etc.
- Provide a public view for a user. It will show the account number and name as well as any other summary information you deem necessary. If the user also acts as a seller, show also email, address, and include a section with all reviews for this seller (see Feedback / Messaging).

Possible additional feature(s):
- Visualize history of balances, spending amounts and purchases by category, etc.

Products / Cart

Basic requirements:
- There is a list of predefined product categories, and each product belongs to one category. At the minimum, a product should have a short name, a longer description, an image, and a price.
- Users can browse and search/filter all products. For each product in the result list, show a summary (e.g., image, name, average review rating, etc.) and link to the detailed product page. At the minimum, support browsing by category, searching by keywords in name/description, and sorting by price.
- A detailed product page will show all details for the product, together with a list of sellers and their current quantities in stock. For each seller, provide an interface for adding a specific quantity of the product from this seller to the user's cart. The cart should also be persistent (i.e. they should remain after the user leaves the site and logs back in). The page should also include a section showing all reviews for this product (see Feedback / Messaging).
- Each line item in the user's cart refers to one product from one seller with a specific quantity. A detailed cart page should list all line items (quantities and unit prices) and the total price. It should also provide ways to change quantities, remove line items, and submit the entire cart as an order. Once an order is placed, the cart becomes empty.
- Users can create new products for sale. The user who created a product will be able to edit the product information.

Possible additional features:
- There are many more enhancements you can make to filtering/sorting a list of products: e.g., sorting by average review rating or total sales, filtering by rating, price, and/or availability of highly rated sellers, etc.
- A flat list of top-level product categories is constraining. Consider a hierarchy of labels, or in general a set of "tags" for labeling products. What is the process for creating and maintaining these labels?
- For the same product, different users in our system may create their own versions of this product in our system. Is there a process for "standardizing" products across sellers, and should we allow different sellers to charge different prices?

Possible additional features:

- Divide the cart into “in cart” and “saved for later” so that a user can check out certain items while keeping the others saved to be purchased at a later time. Items can then be added back into the cart.
- Implement promotional coupon codes for certain items, allowing a user to enter some code to get a discount on either a single item, a group of items, or their entire cart.

Inventory / Order Fulfillment

Basic requirements:

- A user who wishes to act as a seller will have an inventory page that lists all products for sale by this user. There should be a way to add a product to the inventory. For each product in the user’s inventory, the user can view and change the available quantity for sale by this user, or simply remove it altogether from the inventory.
- A seller can browse/search the history of orders fulfilled or to be fulfilled, sorted by in reverse chronological order by default. For each order in this list, show a summary (buyer information including address, date order placed, total amount/number of items, and overall fulfillment status), but do not show information concerning other sellers (recall that an order may involve multiple sellers), and provide a mechanism for marking a line item as fulfilled.
- The entire order should be marked as fulfilled if all line items are fulfilled. Once submitted, an order cannot be changed by the buyer.
- A detailed order page (from the buyer’s perspective) should contain all the information that would have been found on the cart page (see Products / Cart), but with prices “final.” In addition, for each line item, it should show if and when that line item has been fulfilled by the seller.
- A seller should be able to see visualization/analytics about their inventory and order fulfillment pages to see products running low, and popularity / trends of products.

Possible additional feature(s):

- Add analytics about buyers who have worked with this seller, e.g., ratings, number of messages, etc.

Feedback / Messaging

Basic requirements:

- A user can submit a single rating/review for a product. The submission link will be incorporated in the detailed product page (see Products). The user cannot submit multiple ratings/reviews for the same product, but can edit/remove any existing ratings/reviews by this user.
- A user can submit a single rating/review for a seller, provided that the user has ordered something from the seller. Incorporate the submission link in appropriate places in the website, e.g., the detailed order page (see Inventory / Order Fulfillment) and the public view of a seller (see Account / Purchases). Again, the user cannot submit multiple ratings/reviews for the same seller, but an existing rating/review can be edited or removed.

- Each user should be able to list all ratings/reviews authored by this user, sorted in reverse chronological order by default. From this interface the user should be able then select ratings/reviews to update. Incorporate the link to this interface in user account view (see Account / Purchases).
- Produce summary ratings for products and sellers; pages or sections showing lists of reviews for products and sellers. At the very least, the summary needs include the average and number of ratings; the reviews lists should be sorted by rating or date.
- Upvote functionality for rating/reviews to allow certain reviews to be marked as more or less helpful. By default the top 3 most helpful reviews would be shown first, and then the most recent following these.

Possible additional feature(s):

- For an order placed by a user, and for any seller fulfilling this order, the user can start a message thread. This message thread is private to the user and the given seller. Messages should be listed in chronological order, and old messages cannot be edited or deleted. Links to these message threads will be shown in buyers' and sellers' views of orders.
- Include the ability to submit a limited number of images in the reviews, and make these easily available to users on the website.

Putting Them Together

Basic requirements:

- Make sure all functionalities described above under different parts are put together into coherent designs, for both frontend and backend. The website needs to support smooth user task flows across clicks and seamless transitions between different parts.

Possible additional features:

- Develop some methods for recommending products based on past purchase history (both a user's personal history and the history across all users) as well as reviews. The recommendations can be shown in a "You may also like..." section when a user is looking at products, the current cart contents, or even the purchase history.
- In reality, orders don't get fulfilled instantaneously. Design a process by which sellers can indicate when items have been shipped and buyers can verify when items have been received, as well as protocol for resolving issues (i.e., when orders are not fulfilled by a reasonable deadline, when they are returned, or when there are disputes). You might want to rethink when balances are actually updated in your system.

Getting Started & Getting Help

Here is a simple skeleton project implemented using Python, Flask, and PostgreSQL, ready to be up and running. It provides only minimal functionality, however:

- A user can create a profile with the following information:
    - Email address (unique across users)
    - Full name

    - Password
- A user can log in with email and password.
- There is a page listing all items for sale, and users can see the items they purchased previously. However, they come from data preloaded in the underlying database and cannot be changed through the website.

Follow the instructions in README.md to set up your own project.

Duke OIT/CoLab offers a wealth of relevant Roots courses/tutorials/workshops that you might find useful to this project. There are also online resources, such as the official Flask tutorial.

What/When to Submit

See the overall Project Overview (file starting with 1) for general instructions on how to submit. The following spells out the specific contents expected for the standard project option.

Proposal

- README.txt: list your team members, state that you choose the open project option, and pick a short, catchy name for your team. Include in your README.txt a link to our gitlab/github repository, if any (highly recommended!).
- PROPOSAL.pdf:
    - Who is working on which component
    - Any change in outlined design

Midterm Report

1. README.txt: Same as above. Brief instructions on how to run your code.
2. MIDTERM.pdf: showing:
    a. The list of features each of you have implemented or attempted to implement so far. For each feature, indicate its status: fully functional, buggy, partially implemented, or not implemented at all.
    b. Any change in design
3. CODE.zip: all your code.

Final Report

4. README.txt: Same as above. Brief instructions on how to run your code.
5. REPORT.pdf: showing:
    a. Explanation and analysis of the final database schema.
    b. The final list of features you have implemented or attempted to implement (for each member). For each feature, indicate its status: fully functional, buggy, partially implemented, or not implemented at all.

6. CODE.zip: all your code.

Grading Criteria (for 80 out of 100 points)

See the overall Course Project Description (file name starting with 1) for overall project grading criteria and the rest of the 20 out of 100 points. Here we focus on the following score components:

- Assuming individual contributions met perfectly and all 4 parts working together (and proper final report and presentation) meeting the basic requirements will get **a baseline score of 70/80.** Going above and beyond will earn points for your whole team, up to the **maximum score of 80/80.**
- **40 points for individual contributions**
  - Meeting all basic requirements of your part will get a full score. Any buggy or missing features in the basic requirements may lose -0.5 to -8 points, depending on how serious the issues are and based on how many requirements you overall had and fulfilled. You lose 8 points for each requirement totally unfulfilled.
- **20 points for the final product (team score)**
  - **If you lose additional points on top of losing 20 points (ie 0 / 20 for final product), that will be deducted from your individual contributions.**
  - (Integration) For each team member's part that has buggy interaction with other parts or is not integrated into the final website: -0.5 to -5 point, depending on how serious the issues are. You can potentially lose all points for complete lack of integration and teamwork.
  - (Safety Penalty) Some code is susceptible to "SQL injection attacks" (see "optional" slides at the end of Lecture 3): -1 points overall.
  - (Style Penalty) Not following good coding practices (e.g., unnecessarily hard-coding values that should have been obtained from the database; excessive duplication of code without refactoring): -0.5 to -2 points overall, depending on how serious/prevalent the issue is.
  - (Performance Penalty) Unreasonably slow response time that could have been easily fixed (e.g., with appropriate database indexing, or by incrementally receiving and paginating results in the frontend): -0.5 to -3 point overall.
  - (Data Penalty) If your test dataset is too small or totally unrealistic -0.5 to -4 points
  - (Interaction and Aesthetics Bonus) You may recover some of the lost points above (not more than 80 out of 80) for well-designed, smooth user flow interaction, and well-designed and impressive-looking frontend.
  - You will not lose points by going with a plain-looking website coded in plain HTML. After all, this is not a course about web or frontend development!
- **10 points for the final report (team score)**
  - A sloppy or incomplete report may lose -0.5 to -5 points
- **10 points for the final presentation (team score)**

- A visibly careless presentation may lose -0.5 to -5 points