Design Analysis for Shopping Cart Application

Andrew Song (andrew90)

1. Overview

The shopping cart application, as its name already implies, is about "virtual" cart for online shoppers. The goal of this project is to introduce the authentication (Users are allowed to log in, and keep the items in their shopping carts) and authorization (Shoppers and shopkeepers have different roles and actions they can take) through the concepts of cookies and sessions. Another side goal of this project is to actually go through the whole process of software development; The process starts with brainstorming and designing architecture for the software, and eventually leads to actual implementation and finishing. The ultimate objective is to expand the application's functions suitable to more practical usage; these might include refactoring the code for stand-alone shopping cart application, allowing the application in several languages, and so on.

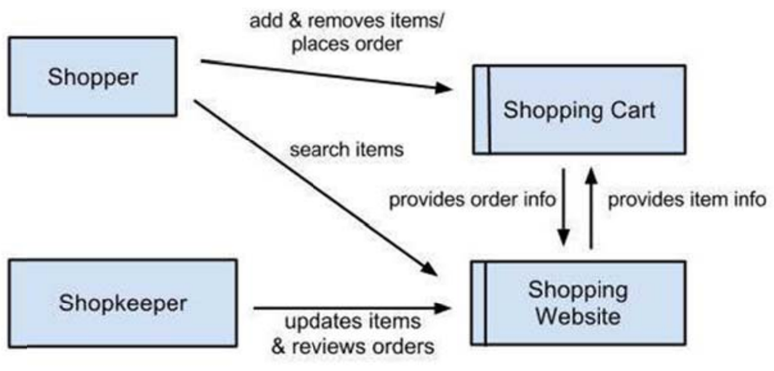


Fig 1. Context Diagram for Shopping cart application. A regular box refers to components external to this application.

Above is the context diagram for the shopping cart application. At this stage of development, the shopping cart application is embedded in the shopping website, although the shopping cart could be refactored to be a stand-alone application, rendering the shopping website as external component. Notice that the shopper does not directly interact with the shopping website; all the interactions with items (except for searching) and placing orders are mediated through shopping cart application.

2. Concept

The key models in the application are User, Cart, Order, and Item. A user logs in as a shopper or a shopkeeper. The shopper is given a shopping cart to where he/she can add and remove items. When the shopper decides to place an order, an order object containing relevant information is created which can be reviewed by the shopper and the shopkeeper in the future. The shopkeeper can modify the existing items, or add new ones. However, since the shopkeeper assumes the role of supplier, he/she will not have access to carts or placing orders. Notice the id fields in the object model diagram; these are important in accessing information.

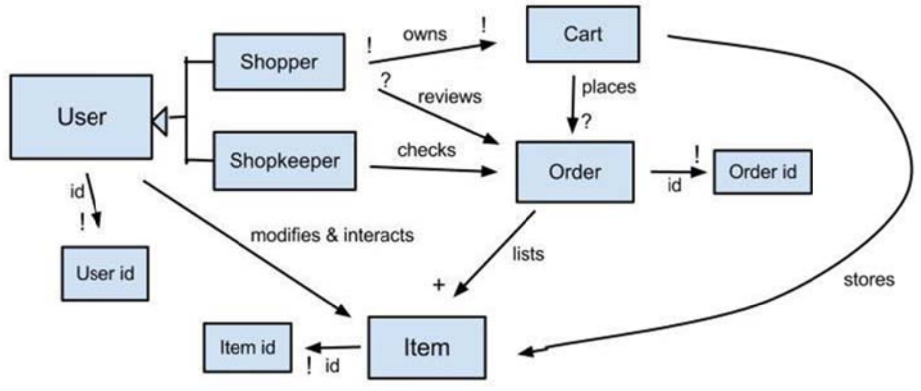


Fig 2. Object Model diagram of Shopping Cart Application

3. Behavior

3.1 Features

Shopper

- **Add & Remove & Update items** : Customer can add, remove, and update items to the cart
- **Place order**: Based on items in the cart, user can place an order.
- **Review orders**: Customers can review the details of their past orders.

Shopkeeper

- **Modify & add new items**: Shopkeeper can modify the description and price of items, as well as adding new ones.
- **Review orders**: Shopkeepers can review what orders have been placed and analyze the purchasing behaviors.

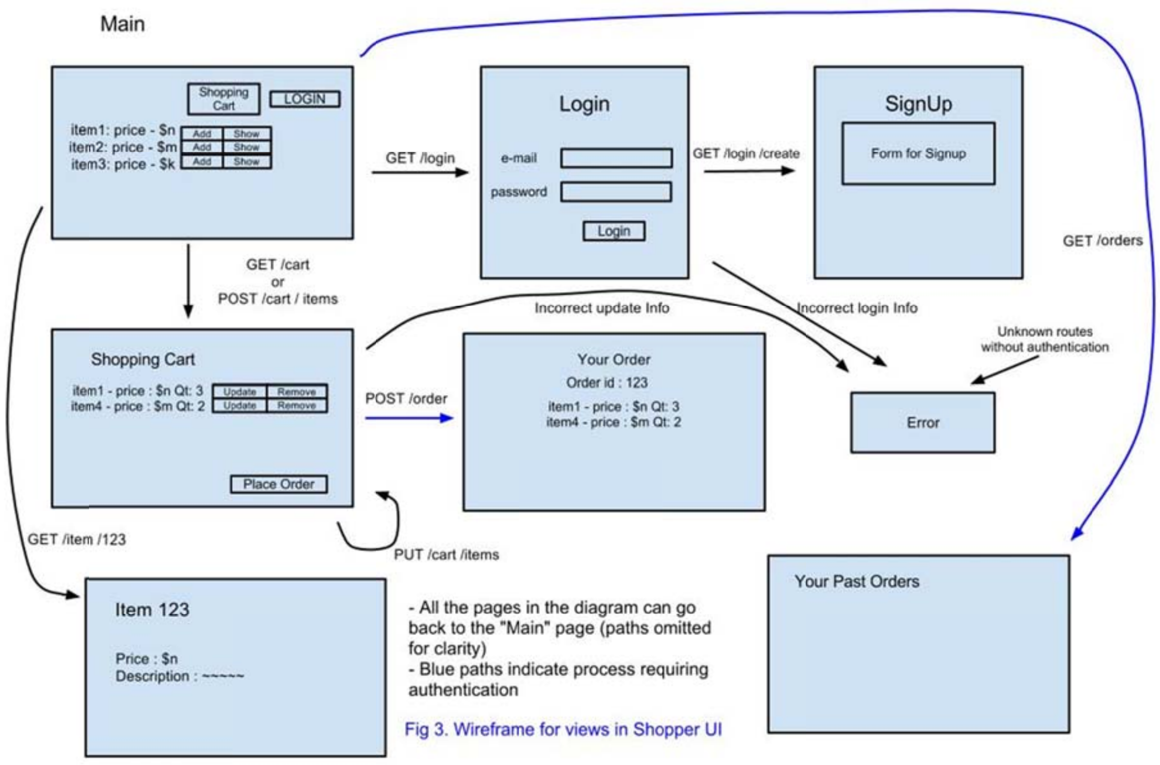
3.2 Security Concerns

The main security threat is session hijacking. If a shopper's session is hijacked by some malicious users, it is likely that orders of unwanted items of unwanted quantities would be placed with the hijacked shopper's credit card information. On the other hand, if a shopkeepers' session is hijacked, existing items might be deleted, or the prices might be reduced to ridiculously cheap levels, allowing the hijacker to place orders at the changed price. Such security threats might be mitigated through following methods (Some are referenced from "Ruby on Rails Security Guide"):

- Provide a secure connection over SSL. This will help prevent so-called "cookie-sniffing" in insecure wireless networks.

- Create a logout button and emphasize the importance of it! Failure to log out in public terminals might cause other malicious users to use the previously logged-in accounts in unwanted ways.
- Not storing sensitive information about users in cookies. Passwords, credit card information, and other security-required information should not be stored in cookies.

3.3 User Interface



- All the pages in the diagram can go back to the "Main" page (paths omitted for clarity)
- Blue paths indicate process requiring authentication

Fig 3. Wireframe for views in Shopper UI

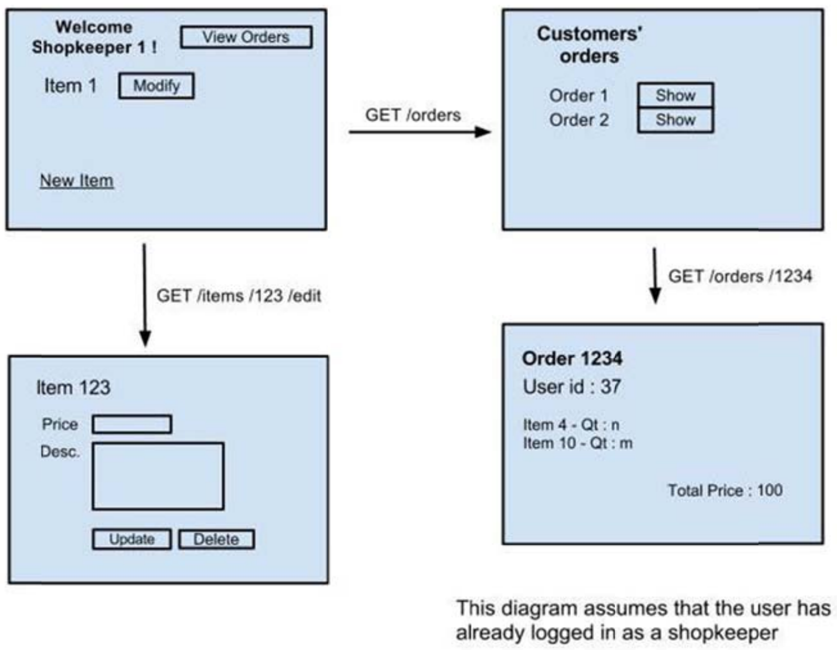


Fig 4. Wireframe for views in Shopkeeper UI

4. Design Challenges

- References to user, cart, order, and items

Constant references to several kinds of objects will be made upon every request from the browser to the server. It is important to store only ids of the objects! Although it is tempting to store all information about items in a cart, such will hinder the performance of the application and fail to update dynamically. For instance, if a shopkeeper changes the price of an item that is already stored in a shoppers' cart, the item in the cart should be updated dynamically. However, if the whole information is stored in the cart, such external changes will be of no effect. Referencing through ids guarantees that such dynamic changes are accorded.

- Store shopping cart items in cookies or sessions?

There are no strict guidelines as to whether the shopping cart items should be stored in browser-side cookies or in server-side sessions. Some research done prior to the write-up has shown that many existing shopping carts differ on such implementations.

For this application, I am using both implementations to handle both cases of shopping cart utilization in not logged-in and logged-in status. If a user tries to add and remove items to the shopping cart without having logged-in as a shopper, cookies would have to record the information, since a session is yet to be created. Once the user is logged-in, a session would have been created, and all the previously cookie-stored information would be transferred to the session. Further actions regarding items would then be stored in the session.

- **Differentiating shoppers and shopkeepers**

Shoppers and shopkeepers have different roles. When accepting a new user (in other words, in sign-in page), the user would be prompted to select the role of shopper or shopkeeper, and such selection would be stored in the User object. In reality, the authentication process of approving a user as a shopkeeper would be much more complicated, but such is not the focus of this project. When a user logs in, different UI will be displayed according to the status of the user.

- **Multiple shopkeepers**

It would be logical to have multiple shopkeepers for a single website instead of just one (Think of it as 3rd party suppliers in the Amazon). Shopkeepers should not be able to modify each other's items, and have access to each other's sales records. Authorization through sessions will be able to limit what the shopkeepers can do and cannot do.

- **Separating Order and Cart model**

"Order" is similar to "cart" in a sense that a user places order based on the items in the cart. However, these two concepts need to be separated. A specific "cart" belonging to a specific user needs to be emptied every time an order is placed, and be prepared for the next order in the prospective future. However, the shopper and the shopkeeper need to be able to look back at the past orders for reference. Therefore, it is crucial to separate two models although they are closely correlated.