PS2 Design Critique

Self-Critique

Summary assessment from user's perspective
Overall, the user interface works well. It is clear how to sell and purchase items, and the divide between the two actions. Having one user login eliminates any confusion as well, given that a user in my design has the capability to both buy and sell items. The navigation through deeper parts of the site is difficult at times, i.e. adding an item requires a long series of steps and clicks, and once a user gets to this point it is unclear how to backtrack. It would be nice to have some sort of indicator for the user to know where they are in terms of the overall page flow. Also, passwords are not hidden both during registration and login.

Summary assessment from developer's perspective
My design allows for differentiations between orders and carts, via separate order items and shop items. This is a strong solution, since shop items are dynamic while order items are immutable. My implementation, however, didn't take advantage of this separation in the model, and as a result, there are still weaknesses in my site. For example, when a storekeeper removes an item, if another storekeeper previously bought it, the item will no longer be reflected in their recent purchases. There are simple fixes for this though, given the strength of my design.

Overall, my specs were good, as was the separation between model and controller; most methods were kept in the model. However, there are a number of variable misnomers that can lead to some confusion. Shopkeeper, the name of my user model, is misleading since shopkeepers can not only sell but make purchases as well. Order items have an attribute called item_number, which refers to the id of the shop item it mirrors. This attribute has a misleading name, as it is unclear whether it refers to a quantity or id. A better name would have been original_id.

Most and least successful decisions
The best design decision I made was to clone items from the cart to the order. This is good because it allows for storekeepers to alter the original item without affecting user's who had already made the purchase, an especially important characteristic for shopping carts used in online stores.

The worst design decision I made was to create and use pointers from the order items to the original store items. While my design allowed for the flexibility described above, my implementation revealed a number of inconsistencies, such that when an item was edited by the storekeeper, these edits were incorrectly reflected in previous orders of that item.

Analysis of design faults in terms of design principles
Using pointers from order items to the original store items is a misrepresentation stemming from my original object model. My object model should have differentiated between the two, since they are in fact separate objects, but instead lumped all items into one object. The mislabeling of item_number and certain methods was the result of not adapting names as the code evolved. The code is obviously dynamic and sometimes it is difficult to keep up with these changes and make sure attribute names properly reflect the underlying design. Finally, the very layered navigation is a result of using a number of different models and controllers. While this is not on its own a fault, there needs to be more transparency between the design and the user

experience.

Priorities for improvement
First priority is fixing the cloning from store item to order item so that more information is stored in the order item, as opposed to just a pointer back to the original. The second is making the navigation through the site more clear, perhaps using some sort of header. The last is to fix attribute and method names to eliminate confusion.

Critique for Malcolm Gilbert

Summary assessment from user's perspective
The added features implemented on your site are really nice and work pretty well, such as the save for later feature and order status. There seems to be some issues with item quantities; when I add an item to my cart twice, there is nothing in my cart that reflects this change, i.e. the item being listed twice or some quantity changing to 2.

Another problem I ran into was when I signed in as a user and then signed in as an admin. My orders seemed to carry over from when I was a user to my admin orders. This was not intuitive to me.

Summary assessment from developer's perspective
Your design documents are very well written, and I had a good sense of your design from those alone. The problem analysis was especially strong. Your specs were similarly clear and thoughtful. There is a good divide between items and items_orders, but you may consider storing more information in items_orders, such that when an admin makes changes to an item, those changes are not reflected in the items_orders object. I was confused by the home controller, since many of those methods seemed like they may belong better in a model. Certain model names were misleading, like shops, which I think should be admin, and devise, which I think should be users.

Most and least successful decisions
The most successful design decision was to remove the shopping cart model. Since your design involved only one storekeeper, you were able to eliminate this entirely and the result is a much simplified model schema.

The least successful design decisions were allowing simultaneous customer and admin logins, not using item quantities, and the implementation of the home controller. As I mentioned earlier, it is confusing for orders to carry over from customer to admin login. While from an admin side, it is definitely justifiable to not use quantities (I guess you can assume there is always stock), from a user perspective, there should be some representation of quantity. Some of the methods in the home controller I think belong in a model.

Analysis of design faults in terms of design principles
Allowing simultaneous customer and admin login is a result of the lack of relationship between shop and customer in the object model. I think that the relationship should be defined as the shop engages with multiple customers, thus not allowing simultaneous login. A different solution may define the relationship differently. The lack of item quantities is also a result of this misdefined relationship. Quantities don't necessarily need to be defined for the admin, but they must be better defined for the customer.

The design of the home controller represents a lack of separation between controller and model. Some of the methods in the home controller belong in the model classes instead.

Priorities for improvement
First, item quantities should have some representation from the user perspective. Second, simultaneous user and admin login should be addressed, either by disabling it entirely or in another way. Lastly, the implementation of the home controllers should be examined and certain methods put in other more appropriate places in the code.