Design Critique for Ibrahim Abdussabur

Summary assessment from user's perspective

Some parts of the UI flow, primarily the checkout process, are a little confusing. On the checkout page, it's ambiguous how there is both a "Checkout" and "Checkout Now" button. Since the "Checkout" button doesn't do anything, I would remove it and put the "Checkout Now" button in its place (below the order summary). Also, after I click "Checkout Now," it brings me to a page with another order summary with another "Checkout" button. Since the shopkeeper interface is updated as soon as I hit "Checkout Now," the cart should probably be emptied at this point (or the order summary should appear as a summary without the "Checkout" button). You might also want to add some kind of message to inform the user that the order has been successfully placed.

The "Search" function brings me to a 404 error page – not sure if you just haven't gotten around to implementing that yet.

It would be helpful to have some way of editing quantities of items placed in the cart other than hitting "Add To Cart" or "Remove" to add or remove one at a time.

On the shopkeeper interface, the quantities for "All Items In Store" are not getting updated when purchases are made, but perhaps you intended to have the shopkeeper manually update the quantity when orders are processed. If that is the case, you might want to consider adding an easy way for the shopkeeper to indicate that an order has been processed (perhaps with a checkbox, dropdown, or something of the sort).

When I try to edit an item or submit a form for adding an item, I get a 500 server error.

The sidebar menu that separates the items into different categories is a nice touch.

Summary assessment from developer's perspective

Nice use of Rails partials to keep the views DRY, and nice separation of methods in models vs. controllers.

In terms of code structure, some improvements could be made according to Rails best practices. For example, you might want to consider having a controller for each model. It might save you some effort to take advantage of Rails inner workings, for example, by using resource routing for a resourceful controller that supports all the CRUD operations just for the Item class.

It might be helpful for both and you and other readers of your codes to sprinkle in a few comments and method signatures.

Most and least successful design decisions

Most successful: use of Rails partials to avoid repeating code in views
Least successful: UI flow of checkout process

Analysis of design faults in terms of design principles

Naming conventions could be a little more streamlined, for example, in /app/models/order.rb you use camel case in one method name and all lowercase with an underscore in the other.

The lumping together of controls under only two controllers may be a design fault (see above), but it may also be a personal preference. In some ways it could be useful to distinguish the shopkeeper controls for the shopper controls.

Priorities for improvement

First, I would look into the shopkeeper interface 500 errors (for add/edit)--these could just be database issues. I would then improve the UI for the checkout process. Disable the search feature if it is not functional to avoid bringing up a 404. My other suggestions are lower priority (more flexible quantity editing, shopkeeper indication of order completion, and controller restructuring).