Team Dexter

12/13/18

Design for Papyr

**Persistent Shopping Cart**

The initial UI design for shopping cart can be seen on page 4 of design sketches. During the planning poker phase, my team came to agreement a persistent shopping cart would be a complex and pervasive task. We discussed how we wanted to handle persistence and decided upon local storage. This decreases network latency since we aren’t storing the cart information in the database. Local storage instead of session storage will allow the shopping cart to remain persistent across tabs and even if we open and close the browser. The shopping cart will require a new checkout component and controller to display the current items in the cart.

**Credit Card Payments**

We knew from the beginning we would be using Stripe to handle credit card payments. Our planning poker led us to believe this would be about the same amount of work as implementing the shopping cart. We will be able to use the same checkout view for the shopping cart. After looking into how stripe works, we can use Stripe elements to create a credit card entry field for checkout. The frontend will request a token from Stripe. This token will be sent to our server along with the checkout information. The server will use request a charge from Stripe using the token. Stripe is doing most of the heavy lifting for us here. The initial UI design for the credit card is part of the shopping cart as well (page 4 design sketches).

**User Profile**

While the application already supports some form of login, by itself, it wasn’t sufficient to provide users with a “personal experience”. We started by ensuring creating a means for users to know that they are indeed logged in. This is done by displaying their username on the top of the application, where they are able to click in to enter their profile. The user profile page displays all information about the user. These information are taken from the server upon login, and stored in the local storage. Users are able to add or remove preferred book categories inside the profile page. Each addition or removal of category on the application sends a request to the server immediately for persistent storage. The User Profile page also allows a user to logout from their account.

**Purchase History**

We decided to push all purchase history into the User Profile page as this seems to be the common practice among several other major e-commerce websites. This design decision was made to reduce the learning curve of using the application. Purchase history is displayed in order for recency, and grouped by purchase orders. The ordering of the purchase history is done by the server. The application’s job is simply to display the purchase history. Each successful checkout stores a purchase history on the server.

**Downloading Purchased ebooks**

Since the client mentioned we could use a dummy pdf (don’t have access to the actual ebooks), this task was rated as non-pervasive and simple. After a quick search I saw you can use HTMLs download tag inside of an href. This user story feature should be a quick addition.

**Custom Experience (Categories)**

We wanted to make the user experience unique to each user and recommend him books based on the preference he has given in his profile. So, for this custom experience feature, once the user is logged in, based on the category preference he has given on his profile, his homepage gets populated with books in these categories. To implement this functionality, we will need to modify homeController.js and homeComponent.js to render the books from user’s categories preference. We can get the category information from the local storage where all user related information will be stored.