Testing

Jake Armendariz

I spent a lot of time working on and testing user sessions and authorization on the backend. For requests that need authentication there are a couple of cases.

- User is not logged in.
- User is logged in as a reader with a session token.
- User is logged in as a publisher with a session token.

Each of these should authenticate the user to access different material. The not logged in user should be blocked from anything except the home and login pages. The reader and publisher should at first glance be blocked from viewing the alternate "manage profile" settings. However, since we authenticate with google, and use email to distinguish users, we can assume that one email can access both accounts. This way, a publisher who also has a reader account, can purchase articles without having to logout and relogin, and a reader who has a publisher account can manage their articles.

Thus, the session checks only need to check for the two equivalence classes of logged in or out, then can return 403 errors on pages where they don't have access, and a 404 error if the user doesn't have an alternate account.

Xin Yan

I tested the working between the front and backend of our our platform. A particular module would be listing the articles associated with each account. The variants include:

- The list of articles owned by a reader.
- The list of articles that are registered by a publisher.

The testing flows are:

- Log in as reader → Purchase article that the reader account can afford → Check reader profile page.
- Log in as publisher → Make a push request to the PayPerRead server to register a new article → Check publisher profile page.
- Log in as publisher → Check publisher profile page → Log in as reader to purchase an article registered by the publisher's email → Refresh and check publisher profile page again.

When a reader purchase an article, the article would be added to the list of articles owned by that reader. To ensure that everything is working as expected, I checked both the database collection and the reader profile page. The reader document's articles field should now have an additional entry with the correct guid of the article that was just purchased. Then, I must also ensure that the reader profile page reflects this update and correctly displays the newly owned article information in the owned articles list.

In order to register an article, the publisher developers must make a push request to our server with the fields stated in our documentation. Once registered, a new entry should be added to the list of registered articles for that publisher, and it should also be reflected in the publisher profile page. When an article is purchased, the view count and total generated revenue of that article should increase accordingly. To test that the logic is functioning correctly, I recorded the previous state of a particular article and proceeded to purchase that article as a reader. Once purchased, I must check that the view count of that article is increased by one, and that the total revenue generated is increased by the articles price. The total revenue generated by an article should equal the its total views multiplied by its price. The total article views of a publisher account should equal the sum of the views in the account's registered articles list.

Daniel Wilby

I devoted time towards testing the sign in/up pages for the reader and publisher. There were a couple different equivalence cases that had to be considered:

- The user did not previously exist
- The user had already made an account

If the user had already existed, log them in and take them to the homepage. If the user hadn't already existed, then take them to a page where they could verify their account creation, and then take them to the homepage.

If the person had made, for example, a publisher account but is signing up for a reader account, then they would fall into the first category. This binary choice made it easy to come up with test cases, trying with an email that had previously made an account, and one that did not have one.

Brendan Teo

The testing I've done involved primarily the frontend component. The module I worked on involves:

- Reader adds balance to account
- Publisher deposits balance into bank

The reader must be able to balance into their account to be able to buy articles. On the other hand, the publisher must be able to make deposits into their account. The testing would include logging either a reader and publisher to be able to perform addition and subtractions to the their balance based on their role as publisher or reader. In other words, if the user is a reader they should be able to view the articles that they bought and add to the balance that they have. For publishers, they should be able to view their articles they published and deposit the revenue they created into their bank accounts through Stripe.