Team 14 - Sprint 3 Retrospective

**Hot TopiX**



**Team Members:** Connor Brown, Hunter David, Steve Eisner, Baxter Govan, Ryan Klinedinst, Victor Pan

**What did we do well?**

**User story #2:** As a user, I would like to be able to delete my account

|  |  |  |  |
| --- | --- | --- | --- |
| # | Description | Estimated time | Owner |
| 1 | Make queries for deleting an account and all of its associated data | 5 Hrs (each) | Hunter,Victor, Connor |
| 2 | Connect User not Found page to deleted accounts | 4 Hrs | Connor |
| 3 | Add a button to delete user’s account from Edit profile page | 2 Hrs | Connor |
| 4 | Testing | 1 Hrs | Connor |

**Completed:** All the functionality for this user story were completed except for the User not Found page. This is because we ended up not implementing the My Profile page, so there was no need for a User not Found page. We also ended up not being able to fully delete users from the MongoDB, but instead using an attribute for checking for deleted users. This is because GetStream doesn’t fully delete users and will end with errors if a new user with the same information is made. In addition, deleted users were not able to log in with their credentials, and they were not able to make a new account with their email associated with the deleted account

**User Story #3:** As a user with disabilities, I would like to be able to easily view the application.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Description | Estimated time | Owner |
| 1 | Make Website Viewable on Mobile | 10 Hrs | Victor |
| 2 | Update Pages with better accessibility | 10 Hrs | Connor |
| 3 | Testing | 1 Hrs | Connor |

**Completed:** We went with a major UI update overhaul from our 2nd sprint. Our sprint 2 had a functional web app, but it did not look the best. We added a lot of Bootstrap and updated UI to make the website viewable on both desktop on mobile. The nav bar was customized so that it would be a dropdown on mobile while a scrolling nav bar on desktop. Connor went through each page and made sure our pages had as close to 100 as possible on each Lighthouse audit.

**User Story #4:** As a user, I want to be able to customize the color palette of my display.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Description | Estimated time | Owner |
| 1 | Create different color schemes for the website | 10 Hrs | Baxter |
| 2 | Implement a way to select different color schemes in the edit profile page | 2 Hrs | Steve |
| 3 | Save the selected color scheme in the database | 5 Hrs | Steve |
| 4 | Add way to update navbar, background, stream, based on the color scheme | 10 Hrs | Victor |
| 5 | Testing | 1 Hrs | Connor |

**Completed:** Baxter ended up making 5 different color schemes for our web app based on Ashwin’s suggestion. We added a way to select the color scheme from the Edit Profile page in a dropdown menu. Hitting the Save button stores the colors for each scheme in the MongoDB database for each user. This way, whenever the user logs back in, the color scheme was saved. We updated most of the UI in terms of color, but we had trouble with the GetStream posts since their React components were already created and not customizable by us. Eventually, Baxter got it working and we were able to dynamically change all the background colors of each page.

**User story #5:** As a user, I would like to be able to follow topics

|  |  |  |  |
| --- | --- | --- | --- |
| # | Description | Estimated time | Owner |
| 1 | Connect topic selection to the stream API | 8 Hrs | Ryan |
| 2 | Build dynamic topic selection | 6 Hrs (each) | Ryan, Baxter |
| 3 | Testing | 1 Hr | Ryan |

**Completed:** Users have the ability to follow existing topics, or add a new topic to the current topics. Unfortunately, filtering by topics does work. A global topic list is stored in MongoDB as well as a topic list for each user. Users can go to the Follow Topics page and see the global list of topics. They can add their own topics as well as follow an existing topic. We added error checks to prevent following nonexisting topics as well as preventing the user from following a topic they already follow.

**User story #6:** As a user, I would like to be able to unfollow topics

|  |  |  |  |
| --- | --- | --- | --- |
| # | Description | Estimated time | Owner |
| 1 | Build topic unfollowing method in server | 6 Hrs | Connor |
| 2 | Implement dynamic topic unfollow page | 5 Hrs (each) | Ryan, Baxter |
| 3 | Implement topic unfollow data wipe in database | 5 Hrs (each) | Ryan, Baxter |
| 4 | Testing | 1 Hr | Ryan |

**Completed:** Users have the ability to unfollow existing topics. We added implementation for users to enter a topic they wish to unfollow. By doing this we were able to remove the requested topic from the users follow information on MongoDB. This is all done by having the users navigate to the Follow Topics page and entering the topic they wish to unfollow.

**What didn’t we do well?**

**User story #1:** As a user, I want to be able to see a user’s profile page with all their posts

|  |  |  |  |
| --- | --- | --- | --- |
| # | Description | Estimated time | Owner |
| 1 | Given a user clicks a user’s name in the timeline, they are forwarded to that user’s profile page | 10 Hrs | Steve |
| 2 | Extend Stream API to show all individual user’s posts on their profile page | 10 Hrs Each | Ryan & Steve |
| 3 | Implement profile page in Navigation Bar | 2 Hrs | Victor |
| 4 | Display a user’s chosen bio at the top of their profile page | 3 Hrs | Steve |
| 5 | Display a user’s profile picture at the top of their profile page | 5 Hrs | Steve |
| 6 | Testing | 1 Hr | Hunter |

**Not Completed:** Since we used the stream API only on the client side in React and not in our backend, we could not retrieve information about the current post, user, etc in pages other than the stream page. This caused many complications and difficulties when attempting to create individual profile pages for users. Retrieving posts from a user required knowing their username, which we were unable to get from our API. Similarly, this prevented us from implementing sorting by topics as well, since the topics added to a post could not be retrieved after posting.

We tried three different solutions to solve the problem with creating profile pages. The first was to implement the profile page in HTML and render it server side, like how we made our follow user pages or our edit profile pages. The problem we saw here was that there was no way to forward a user to this page from the client, i.e. there was no way they could click a button in the timeline and go to the given user’s profile. Realizing this we then tried to render the profile page using react on the client side. We saw complications here because there was no way to retrieve user (username, bio, etc.) info from the timeline. Lastly, we tried a hybrid approach using EJS and react javascript embedded within an HTML page being rendered by the server. This approach saw more success, but we still couldn’t route users through the timeline page or display their individual posts.

**What can we improve on?**

Once again we were over-ambitious with user stories. Ashwin asked us to add a few more user stories, but we overestimated the amount of work it would take. This forced us to revise our sprint planning document. We could improve on better estimation of work and planning work per sprint a lot better. Overall, we were able to create a functional web application that satisfied most of our goals.

We think that one of our biggest challenges with this sprint was having to use the getstream API. This API was excellent when it came to getting through sprints 1 and 2 and basic social media functionality. However, for some of the more niche functionality we wanted to implement during this sprint, many aspects of the getstream api made this more difficult. For example, getstream react elements are nearly impossible to edit and therefore scored lower on the lighthouse accessibility test than the elements we implemented ourselves. In the future, we would rather make slower progress early on and not use an api like this in order to have more control over every aspect of our project.