Purvi Sehgal Reflection

Implementing the unread and read messages views were among one of the hardest features to implement. I spent around 7 hours on it because it caused many bugs. It was difficult because once the message had been clicked, it needed to be removed from the page, but it couldn’t be removed from the server since the read message view was still fetching data from the server. The problem with keeping it on the server was that when the sort by button was pressed and the platform was selected, it would populate the read messages on the unread messages page (since it was still on the server). I got around this by implementing a key in localStorage with the information and checking if the message is in localStoarge (and hence read) before populating messages on the unread message page. Another challenging aspect of the feature was changing from the read message view to the message history view when the message button was selected. I addressed this by making my functions generic. For example, I made my toggleView function generic such that it takes in a start view and an end view. One of the most challenging bugs was from implementing this feature. When the read message view was displayed, it added an event listener, so the next time the read messages list would be populated, it would add extra elements. I resolved the issue by removing event listeners when necessary.

One unsuccessful feature was utilizing third party API’s to populate messages and chat information. I attempted for at least 10 hours with different platforms – Facebook, LinkedIn, Discord, and Twitter, but they didn’t work for different reasons. For example, for Facebook, after resolving multiple errors especially permission and scope related issues, I was able to redirect users to an OAuth page, allow them to sign in, get an identifier code from them, implement a successful token exchange (another api call) to get the token from the code. However, after all of this, I found out that Facebook doesn’t allow accessing the read messages scope to non-business accounts. I knew that switching to a business account for Instagram took under a few minutes, so I thought the process would be very similar for Facebook since they are owned by the same company. However, for a business Facebook account, there was an entire verification process, which would require showing documents for an established business. Then, I switched to using the LinkedIn API, but for some reason, there were scope and permission related issues. There wasn’t much available online, and I watched all of the videos I could, but this was all in vain. I attempted another platform, Discord, knowing that getting messages was free and could be achieved with a basic developer account. Again, I was able to redirect users to an OAuth page, allow them to sign in, get an identifier code from them, implement a successful token exchange (another api call) to get the token from the code. I got further in this, and I knew that it was successfully listening to messages from channels (based on print messages and debugging). However, it wasn’t able to access any of the messages. After researching, I concluded that there were permission related issues for the channels I was trying to pull information from. I attempted multiple fixes, but they didn’t work. Lastly, I tried using the Twitter API, but while it was very well documented, there was an issue with that too.

Overall, despite the issues, I’ve really enjoyed this project, and it has helped me grow a lot as a programmer. Plus, the challenges made the project even more exciting to implement.