****

**Probos**

Team 14 (Team Mayhem) - Sprint 2 Retrospective

*Alexa Baker, Daniel Raftery, Steven Daniel Vaughan, and Michael Woods*

**What went well?**

Overall, while media and notifications had their problems, most aspects of the sprint went well. In regards to a problem with server calls delaying UI display, an overhaul of the threading system implementation had astonishing effects and drastically reduced the amount of time in which the UI thread was hung up. This change in threading, along with a few additional fixes, also led to resolution of the bugs that were still lurking from the first sprint. The addition of the mute button, privacy choices for messages, and the ability to change basic profile settings finished the most basic parts of Mastodon client functionality and will allow us to start expanding our program to be unique from competing apps. Adding the option to redraft and/or delete messages marked the first addition to the app that is not present in some competing Mastodon apps, such as Tusky. The saved drafts presented a problem since there was no API call for this action. This led to the development of a creative in-app solution that does not involve the server.

**User Story 2**

I want to change the frequency push notifications are generated.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Create UI page to change notification settings, access through a button visible from the timelines | 4 hours | Alexa |

Completed:

* Using the common background color and the foundation of the timeline UI for the structure of the UI, I added 4 switch buttons to switch certain types of notifications on and off that can be easily connected to the notification options when they are enabled.

**User Story 3**

As a user, I want to mute/block other users.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Add mute/block button on user profile UI in a design consistent manner with follow button | 1 hour | Daniel |
| 2 | Researching API function possibilities for muting/blocking | 1 hour | Daniel |
| 3 | Implement API functionality to button for blocking and muting | 2 hours | Daniel |
| 4 | Implement storage of blocks/mutes | 3 hours | Daniel |
| 5 | Add confirmation screen asking after button press if the user wants to block the user | 1 hour | Daniel |
| 6 | Test muting/blocking functionality, fixing bugs | 2 hours | Daniel |

Completed:

* We found a way to mute/block users without using locally stored lists, so it was possible to implement those features in a more efficient manner, essentially improving upon the sprint’s plan. We added a button on user statuses which, when clicked, provides a confirmation screen clarifying what the user is about to do. If the “yes” on the confirmation screen is clicked, the user who wrote the status that was clicked will be muted and no longer appear on the timeline after refreshing.

**User Story 4**

As a user, I want to change my display name.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 2 | Update code to retrieve text from the field correctly to parse into API calls. | 2 hours | Michael |
| 3 | Test through both app and web client that display name update is processed correctly. | 1 hour | Michael |

Completed:

* The user can find an option to update their display name in the dropdown menu at the top of the timeline views. When tapped, the user is presented with a popup message containing an editable text field in which a new name can be entered. Upon confirming the entered text, an API call is made to the instance to update the user’s account. Upon canceling, no update request is sent.

**User Story 5**

As a user, I want to edit my account “bio” information for a specific instance.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Update “bio” field code in the editable profile UI to retrieve text correctly and create the appropriate API call. | 2 hours | Michael |
| 4 | Test through both app and web client that the “bio” update is processed correctly. | 1 hour | Michael |

Completed:

* The user can find an option to update their bio in the dropdown menu at the top of the timeline views. When tapped, the user is presented with a popup message containing an editable text field in which text can be entered. Upon confirming the entered text, an API call is made to the instance to update the user’s account. Upon canceling, no update request is sent.

**User Story 7**

As a user, I want to save message drafts.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Implement a save button in the message draft UI | 1 hour | Daniel |
| 2 | Design a method for storing drafts locally, or saving them to the instance | 3 hours | Daniel |
| 3 | Implement a button to access saved previous drafts and replies, in a layout consistent with the design of the rest of Probos | 1 hour | Daniel |
| 4 | Implement a UI to view a list of previously saved drafts, upon tapping any previous draft the draft/reply UI is loaded in order to continue writing before posting | 4 hours | Daniel |
| 5 | Connect buttons to correct API calls/Implement retrieving drafts from Mastodon instance | 3 hours | Daniel |
| 6 | Test functionality of saving drafts after app closes, user logs out, etc. | 2 hours | Daniel |

Completed:

* This was implemented with three locally stored persistent “save slots” for drafts with which a user can save and load their previously typed messages. To access the feature, the user taps on the button in the draft UI with the save icon, and then chooses between two menu options: Save or Load draft. Once one of the two is chosen, a second menu appears with three options for corresponding save slots. Users can use this to send statuses without worrying about losing their draft if they cannot finish it, or would like to revise it before posting.

**User Story 8**

As a user, I want to be able to delete and redraft my previous messages.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Implement a Delete button in the “more” dropdown on status UI on statuses you have written, and create confirmation screen asking for affirmation of delete request | 1 hour | Daniel |
| 2 | Research API on calls to delete posts | 1 hour | Daniel |
| 3 | Implement deleting posts | 1 hour | Daniel |
| 4 | Implement a Redraft/Edit button to the “more” dropdown on status UI on statuses you have written, and send to a modified version of draft UI containing the previous message | 2 hours | Daniel |
| 5 | Research API on replacing status content and implement functionality for replacing statuses | 3 hours | Daniel |
| 6 | Test functionality of redraft with logging out, media statuses, text statuses, etc. | 2 hours | Daniel |

Completed:

* We found a way to implement a delete button that is only visible if you are the author of the post, simplifying the UI to keep a consistent look with the other features. Once a user finds a status of theirs they would like to delete, they tap the button and a menu screen appears with two options: Delete, and Delete and Redraft. If Delete is selected, a confirmation screen appears. If the “yes” button is pushed on the confirmation, the status is deleted.
* The other option is when the user pressed the “Delete and Redraft” selection in the delete button’s menu, which loads a specialized version of the draft UI containing the text from the status previously.

**User Story 9**

As a user, I want to attach a content warning to “toots” as needed.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Update UI for messages to display content warnings and have message text/media hidden | 4 hours | Steven |
| 2 | Reimplement UI’s to include buttons suggesting statuses data on content warnings | 3 hours | Steven |
| 3 | Research API on methods involving content warnings/hiding | 2.5 hours | Steven |

Completed:

* When a message is being processed for display, if it is marked as sensitive, any media attached is not shown by default, and red text for the content warning takes the place of the actual message text. A Show/Hide button was added to allow for the message text and media (if any) to be toggled into display state or back to hidden.

**User Story 10**

As a user, I want to show or hide media attached to a “toot.”

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Update message UI to display media attached to a message (if any). | 4 hours | Michael |
| 2 | Add ability to hide or show message media, depending on its state. | 2 hours | Michael |
| 3 | Implement API call/check for whether a message’s media is sensitive or not to set default display state. | 2 hours | Michael |
| 4 | Test that displays state toggles correctly and is defaulted properly for the message’s sensitivity state. | 1 hour | Michael |

Completed:

* When messages are displayed in the timeline, if media is not contained within one, that status item’s ImageView is not displayed and the message appears as a normal text status. If there is an image attachment to the status, the ImageView is set to display and the status resizes as needed to accommodate it. However, this is superseded by message sensitivity state, and the status will not display at all if marked as such. The Show/Hide button at the bottom left of each status toggles the message between the display and hidden states, showing the content warning (if any) in place of the message text.

**User Story 11**

As a user, I want to adjust the privacy setting of a “toot” (public, unlisted, followers only, direct message) on a message-by-message basis.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Update draft/reply UI to include options for privacy settings on potential messages | 2 hours | Daniel |
| 2 | Research API to know how to change visibility settings of a status | 1 hour | Daniel |
| 3 | Update functionality of draft/reply to include privacy data on outgoing status posts | 3 hours | Daniel |
| 4 | Implement functionality to change privacy settings of a status through edit/redraft functionality | 2 hours | Daniel |
| 5 | Test functionality of privacy settings on new timelines, statuses, from users perspective, web perspective, another users perspective | 2 hours | Daniel |

Completed:

* When a user is drafting a new status, there is now a button in the bottom left that when pressed displays a menu for different visibility settings for that post. The default setting for the post when the menu is not accessed is public, with other options being Direct, Unlisted, and Private.

**Now-Complete User Story 12**

As a user, I want to be able to follow other users.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 3 | Test that profile information loads correctly and that following (or unfollowing) works correctly | 2 hours | Alexa |

Completed:

* With the help of Daniel’s fix to the boost and reblog button, I used the same fix involving the addition of a local boolean to track follow/unfollow status, since the API would not update this boolean until the timeline was refreshed. Moving the function to a new location with the threading changes alongside this fixed the issue.

**More-Complete User Story 13**

As a user, I want to attach media to a “toot.”

Tasks Required

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Add a button to the message draft UI to attach media to a “toot.” | 1 hour | Michael |
| 2 | Attach a system call to the UI button to open the media picker. | 3 hours | Michael |
| 3 | Ensure that API calls are updated (if need be) to account for sending the attachment. | 2 hours | Michael |

Completed:

* When the user begins drafting a message, there is a button available to select media attachments. When selected by the user, a media picker is opened which allows for a single file to be selected. If the selection is canceled, nothing is changed about the draft. If an image is chosen, its file path is parsed into the proper format and handled by the activity when the message is submitted to the instance.

**Developer Stories**

As a developer, I want the thread execution to be optimized and to not interfere with UI activities.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Testing thread execution with debugging | 3 hours | Steven |
| 2 | Researching thread interactions with UI in Android | 3 hours | Steven |
| 3 | Testing alternative execution methods | 3.5 hours | Steven |
| 4 | Implementing more efficient execution based on the activity being done | 3 hours | Steven |

Completed:

* A new thread execution method was implemented, and this allowed the UI thread to continue execution without ever being hung up. This makes code execution in the app much more efficient and improves the flow of the app’s UI.

As a developer, I want to fix the bug where messages that did not have a favorite/”boost” applied to them would appear with one.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Testing button functionality with debugging | 3 hours | Steven |
| 2 | Researching better methods of storing user boosts/favorites client-side | 3 hours | Steven |
| 3 | Implementing more efficient storage methods and API calls | 3 hours | Steven |

Completed:

* The bug that caused some statuses/”messages” to display that their favorite and/or boost buttons were active was identified and corrected. The buttons now function as intended, with only messages that were favorited or boosted displaying the corresponding button as active.

**What did not go well?**

**In general,** we encountered many issues related to the time required to implement functionality for our application, given that all team members are also enrolled in CS252 and had to work on the shell lab for the first two weeks of the sprint. Additionally, while media selection itself was sorted out, there was little to no documentation on how to submit such media to the instance, and each attempt resulted in errors. On top of all this, we did not realize beforehand that implementing notifications would require creating and running a secondary service to handle pushing them.

**User Story 1**

As a user, I want to choose what types of notifications I receive from Probos.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Set notifications for direct messages, mentions, and replies | 4 hours | Alexa |
| 2 | Set notifications for follows, boosts, and favorites (general notifications already figured out) | 2 hours | Alexa |
| 3 | Implement automatic refreshing of notifications when Probos is in the background | 3 hours | Alexa |
| 4 | Implement tapping on notifications to bring you to the relevant toot/message | 4 hours | Alexa |
| 5 | Test various edge cases highlighted in the testing document | 2 hours | Alexa |

Not Completed:

* Notifications, while not only non-existent in our normal API, apparently have been a problem for many app developers, since Mastodon originated online and only natively supports web-based notifications. While there are a few ways to implement notifications on Android and there have been some systems to support other Mastodon apps, it was discovered that to do this we must create a completely separate program to get notifications from a server and have that program push notifications to Probos.

**User Story 2**

I want to change the frequency push notifications are generated.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 2 | Connect UI to code to enable and disable types of notifications | 3 hours | Alexa |
| 3 | Test notification settings and various edge cases highlighted in the testing document | 3 hours | Alexa |

Not Completed:

* While the UI is created and can be navigated to in the app without a functioning notification system (now most likely a functioning extra program), even if calls were implemented there would be no way to test if they worked. Overall, without a better understanding of notifications these could not be implemented anyways.

**User Story 4**

As a user, I want to change my display name.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Create a duplicate, editable version of the profile UI for the user to interact with. | 3 hours | Michael |

Not Completed:

* While the functionality was implemented in a very simple form, the editable profile UI was not created to do so as a time-saving measure. This can be done later to tie functionality such as this and updating the user bio into a single activity.

**User Story 5**

As a user, I want to edit my account “bio” information for a specific instance.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 2 | Update the user profile UI to include a variable number of “metadata” fields. | 4 hours | Michael |
| 3 | Update “metadata” fields in editable profile UI to retrieve text correctly and create the appropriate API call(s). | 2 hours | Michael |

Not Completed:

* While the ability for a user to update their bio information was implemented in a very simple form, the user profile UI was not updated to accommodate user metadata fields, and the editable profile UI has not yet been created.

**User Story 6**

As a user, I want to edit my profile icon and banner images in a specific instance.

Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Update code connected to both user profile and user banner images on the editable profile UI to open a media picker. | 3 hours | Michael |
| 2 | Ensure the media file selected is of a valid type to use for a Mastodon profile. | 3 hours | Michael |
| 3 | Generate the appropriate API call with the now appropriately-formatted images. | 2 hours | Michael |
| 4 | Test through both app and web client that the profile icon and banner image updates were processed correctly. | 1 hour | Michael |

Not Completed:

* Besides the issue of the editable user profile UI not yet being implemented, we have yet to determine the exact required formatting for submission of media to an instance, so while we have determined how to access the system media picker for image selection, we cannot make use of that knowledge yet and have not applied it to this functionality.

**User Story 9**

As a user, I want to attach a content warning to “toots” as needed.

Tasks Required

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 4 | Test functionality with redrafting content warnings, general bugs scrolling new messages | 2.5 hours | Steven |

Not Completed:

* While the other tasks were completed as a prerequisite to this remaining one, we simply did not reach it with enough time for completion. However, implementing the functionality to add a content warning to a message should not be exceedingly difficult and can be taken care of early in the next sprint.

**Still-Incomplete User Story 13**

As a user, I want to attach media to a “toot.”

Tasks Required

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 4 | Test that messages appear correctly locally and remotely. | 2 hours | Michael |

Not Completed:

* While media selection works properly, the submission format required by the instance in order to post the media could not be determined due to a lack of documentation in the API, and as a result all current attempts to post media through the app result in a failed attachment. Statuses still show as having an attachment, but one that cannot be found.

**Still-Incomplete User Story 14**

As a user, I want to view “toot” messages sent by others.

Tasks Required

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 5 | Build UI for the expanded view of a “toot” | 2 hours | Steven |

Not Completed:

* While the expanded view was expanded upon with further details, such as the number of favorites and boosts, it is still unable to display previous and subsequent messages (if any) in addition to the expanded status, so this story remains unfinished.

**How should you improve?**

Our first and foremost priority for Sprint 3 is prioritizing development in a way that reduces the amount of time that user stories are dependent on others so that there is less time to wait before being implemented. The best ways to achieve this involve ranking each user story in terms of which needs to be implemented first, and splitting user stories of different ranks between different planned weeks. This helps remove stalls in our development pipeline and will hopefully make a significant difference in the future.

While we improved in our documentation, another thing we can improve upon is writing test cases. We will look into writing automated test cases and their feasibility with our project, which will hopefully streamline our testing times and presentations by a wide margin. We can also do a better job at recording any manual testing that we complete for the final review for the cases that are more difficult to automate.

Although this sprint is longer than the other two, it would be a mistake to overestimate our abilities and not be able to deliver on our goals for the final sprint. It would be extremely helpful to ensure that for this final sprint that we do not overwhelm ourselves with the amount of work required by taking other coursework into better account. Our estimates for the required time were more accurate for this sprint, but that does not help if team members are preoccupied with other responsibilities and obligations and unable to put time towards their features.

Some problems we encountered over Spring Break involved conflicting schedules; as a result, we had to rely on remote communication with each other in order to continue work on Probos. This can be improved in the future by telling everyone when they are available when the team cannot meet in person, and thinking ahead of what information other team members might need when they are not available. The last thing any of us needs is a communication problem that forces us to wait when we are in the swing of things, as jumping back into half-implemented features takes much longer and is generally advisable to avoid at all costs.

Finally, it is crucial that we are fully prepared for the final sprint presentation, more so than previous presentations. Taking this into account, our team should do better with wrapping up any new development at least three days before the demonstration, leaving plenty of time to prepare our presentation materials and sufficiently rehearse our demo.

exit(0);