****

**Probos**

Team 14 - Sprint 1 Planning Document

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

**Sprint Overview**

During this sprint, we hope to lay a functioning groundwork of user interface and core Mastodon functions while we acquaint ourselves with the use of the API. This includes user login and the subsequent retrieval of information from the user’s instance and simple actions such as sending a “toot” message.

**Scrum Master:** Michael Woods

**Meeting Plan:** Mondays/Wednesdays @ 7:00 PM

**Risks and Challenges**

In addition to learning the Mastodon API from the ground up, we will need to ensure that our project follows the service’s application guidelines at every step. We will also need to ensure each of us correctly sets up GitHub integration with our installations of Android Studio in order to keep up-to-date with the most recent developments. Additionally, we must remember to code in such a way that the base features we implement now can be reused, expanded on, and brought together with relative ease in later sprints.

**Current Sprint Detail**

1. As a user, I want to log in to my account(s).
   1. Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Build UI for the login page | 2 hours | Steven |
| 2 | Connect data fields and buttons to functions/API calls | 2 hours | Steven |
| 3 | Test API calls to the server using the login page | 2 hours | Steven |
| 4 | Implement saved credentials for less frequent login requirements | 2 hours | Alexa |

Acceptance Criteria:

* Given that the UI is implemented correctly, text entered by the user should not distort the size of the field and instead be left/right scrollable.
* Given that the API calls are made correctly, when incorrect login information is provided the app should return an error message to the user and prompt them again.
* Given that the user has previously logged in successfully and not logged out since then when the app launches, it should pass over the login UI entirely.

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

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Build UI for Timelines | 2 hours | Steven |
| 2 | Retrieve messages via API call(s) for display within the UI | 2 hours | Daniel |
| 3 | Implement a method of refreshing the message list | 2 hours | Daniel |
| 4 | Test Timeline UI functionality and API calls | 2 hours | Michael |
| 5 | Build UI for the expanded view of a “toot” | 2 hours | Steven |

Acceptance Criteria:

* Given the timeline UI is implemented correctly, when the user navigates to a timeline, the messages should be displayed in reverse chronological order from the top of the page to the bottom.
* Given the timeline refresh is implemented correctly, when the user requests a refresh, new messages should be appended to the top of the UI list without totally re-retrieving the timeline or losing the user’s position in the list
* Given the expanded message UI is implemented correctly, when the user taps to open a specific “toot,” the app should display previous and subsequent messages (if any) at regular message size in chronological order above/below the expanded message.

1. As a user, I want to send “toots.”
   1. Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Build UI for drafting “toots” | 2 hours | Daniel |
| 2 | Connect data fields and timestamp to functions/API calls | 2 hours | Daniel |
| 3 | Test API calls to the server using draft page | 2 hours | Alexa |
| 4 | Include refresh on message list to include new toot | 2 hours | Daniel |

Acceptance Criteria:

* Given the UI for drafting a “toot” is implemented correctly, when the user enters text beyond the display size of the message field, the field should become up/down scrollable.
* Given the API calls for drafting a “toot” are used correctly, when the user presses the back arrow either on-screen or on-device, the app should prompt the user if they would like to save the message as a draft.
* Given the API calls for drafting a “toot” are used correctly, when the user presses the send button, their message should be processed and posted and viewable according to the selected message privacy setting.

1. As a user, I want to be able to log out of or switch the account I am using.
   1. Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Add button in menu UI for logout and return to login UI | 1 hour | Steven |
| 2 | Connect button to API call and remove saved credentials | 1 hour | Alexa |
| 3 | Test API calls to the server using profile page and check that login credentials are removed | 1 hour | Alexa |

Acceptance Criteria:

* Given the API call is connected and used correctly, when the user chooses to log out, that account’s credentials should be removed from the application.
* Given the API call is connected and used correctly, the user should be returned to the login page.
* Given that there are multiple accounts logged in on the app, when the user logs out of an account, the app should switch to another logged-in account’s home page when they log out.

1. As a user, I want to be able to follow other users.
   1. Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Build user profile UI | 4 hours | Steven |
| 2 | Connect UI to API calls for both functions and data retrieval | 3 hours | Alexa |
| 3 | Test that profile information loads correctly and that following (or unfollowing) works correctly | 2 hours | Alexa |

Acceptance Criteria:

* Given that the UI is configured correctly, when the user presses on a profile picture/name, the corresponding profile page should open.
* Given that the API calls are connected and used correctly, the user should be able to see profile information on the profile page.
* Given that the API calls are connected and the UI is configured correctly, the user should be able to press the various buttons on the page (“follow/unfollow”, ”block”, etc.) the corresponding action will be completed.

1. As a user, I want to see my list of followers.
   1. Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Create a UI similar to a timeline to display users in a list you can scroll through (around 50 users initially displayed) | 2 hours | Steven |
| 2 | Connect UI to API calls to retrieve the following list | 2 hours | Michael |
| 3 | Add option to sort the list of users (default, alphabetical ascending/descending) | 2 hours | Michael |
| 4 | Test API calls and sorting options using the following page | 2 hours | Michael |

Acceptance Criteria:

* Given that UI is configured correctly, the follower list should appear in the correct format.
* Given that the follower UI is configured correctly, the follower list should scroll correctly(same as messages)
* Given the sorting button works, the follower list should stable sort to the selected option

1. As a user, I want to scroll to view older “toots”.
   1. Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Ensure the timeline UI is able to adjust to accommodate additional messages. (50 at a time) | 3 hours | Steven |
| 2 | Create a function to retrieve additional messages via API and expand the timeline downwards | 2 hours | Michael |
| 3 | Test that additional messages maintain correct order when appended to the end of the list | 2 hours | Michael |

Acceptance Criteria:

* Given that the UI is configured correctly, when the list of messages is loaded for the first time, it will initially contain the 50 most recent messages.
* Given that the API is configured correctly, the user will be able to scroll down to view more than the 50 most recent messages.
* Given that the API is configured correctly, the oldest age of viewable messages will be ~24 hours.

1. As a user, I want to favorite “toots.”
   1. Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Ensure a button to favorite a “toot” is available in both timeline view and expanded view messages. | 2 hours | Daniel |
| 2 | Connect the UI button to an API call to mark the message as favorited (or unfavorited). | 2 hours | Alexa |
| 3 | Test with web client that messages are favorited (or unfavorited) appropriately. | 1 hour | Alexa |

Acceptance Criteria:

* Given that the favorite button works, one tap changes the image/color of the favorite object and update number of favorites by one higher in the expanded view.
* Given that the favorite button works, a second tap on the icon should change it back to the original color/image and update number of favorites by one lower in the expanded view.
* Given that the API has a successful favorite call, the numbers of favorites in the expanded view should update for a non-local user.

1. As a user, I want to reply to specific “toots.”
   1. Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Ensure a button to reply to a specific “toot” is available in both timeline view and expanded view messages that opens message composition. | 2 hours | Daniel |
| 2 | Connect the UI button to API calls to get conversation identifiers to maintain conversation threading. | 2 hours | Daniel |
| 3 | Implement a modified toot draft functionality/screen into the reply message field. | 2 hours | Daniel |
| 4 | Test button and toot functionally in different scenarios (timeline view, conversation view, etc) | 1 hour | Michael |

Acceptance Criteria:

* Given that the UI is configured correctly with underlying function calls, when the user presses the reply button in either timeline or expanded view, functionality will remain the same.
* Given that the API is connected and used correctly, the reply button will successfully post a message at the correct point in the conversation.
* Given that the UI is configured correctly, when the user chooses to cancel a message and save it as a draft, it still maintains the identifier to what message it would be in response to.

1. As a user, I want to boost “toots.”
   1. Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Ensure that there is a button available in both timeline view and the expanded view to boost (or unboost) a given “toot.” | 2 hours | Daniel |
| 2 | Connect the UI button to an API call to mark the message as boosted (or unboosted). | 2 hours | Alexa |
| 3 | Test with web client that messages are boosted (or unboosted) appropriately. | 1 hour | Alexa |

Acceptance Criteria:

* Given that the boost button works, one tap change the image/color of the favorite object and update number of boosts by one higher in the expanded view.
* Given that the boost button works, a second tap on the icon should change it back to the original color/image and update number of boosts by one lower in the expanded view.
* Given that the API has a successful boost call, the number of boosts in the expanded view should update for a non-local user.

1. As a user, I want to attach media attached to a “toot.”
   1. 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 the attachment. | 2 hours | Michael |
| 4 | Test messages appear correctly locally and non-locally | 2 hours | Michael |

Acceptance Criteria:

* Given that the media picker is called to properly, if a file is of an incorrect type, then the user should be unable to select it.
* Given that the media picker is called to properly, if a file is too large to submit, then it should not be attached and the user should be notified as such.
* Given that the API calls for submitting media are used correctly, when a user submits a “toot” with media, it should be processed correctly and viewable from other clients.

1. As a user, I want to change which timeline I am looking at (personal, local, or federated).
   1. Tasks Required:

|  |  |  |  |
| --- | --- | --- | --- |
| Order: | Task: | Estimated Time: | Assigned To: |
| 1 | Ensure the main UI includes three tabs, one for each of the timelines. | 2 hours | Michael |
| 2 | Maintain separate message lists and timeline positions for each tab. | 3 hours | Alexa |
| 3 | Update API calls for each tab to ensure they refresh with the appropriate messages. | 1 hour | Michael |

Acceptance Criteria:

* Given that the UI is configured correctly, when the user launches the app, all three timelines should load independently.
* Given that the UI is configured correctly, when the user refreshes a timeline, the other two should remain in their current state.
* Given that the UI is configured correctly, when the user taps on a different timeline’s tab, the message list transitions correctly and brings them to their previous position in the list (if previously viewed).

**Remaining Backlog:**

Functional Requirements

1. **Accounts**

As a user,

* 1. I want to register an account on a specific instance.
  2. ~~I want to log in to my account(s).~~
  3. ~~I want to be able to log out of or switch the account I am using.~~
  4. I want to be able to go through my instance’s process for resetting passwords.

1. **Messaging in the Timeline / “Toots”**

As a user,

* 1. ~~I want to view “toot” messages sent by others.~~
  2. ~~I want to scroll to view older “toots”.~~
  3. I want to view a list of the “toots” I have favorited.
  4. ~~I want to favorite “toots”~~
  5. ~~I want to reply to specific “toots”~~
  6. ~~I want to boost “toots”.~~

1. **Creating a Message/ “Toot”**As a user,
   1. ~~I want to send “toots.~~
   2. I want to adjust the privacy setting of a “toot” (public, unlisted, followers only, direct message) on a message-by-message basis.
   3. I want to show or hide media attached to a “toot.”
   4. I want to attach a content warning to “toots” as needed.
   5. I want to be able to delete and redraft my previous messages.
   6. I want to save message drafts.
2. **Instance Settings**

As a user,

* 1. ~~I want to change which timeline I am looking at (personal, local, or federated).~~
  2. I want to carry over the list of profiles I follow to a new account on a different instance.
  3. I want to edit my profile icon and banner images in a specific instance.
  4. I want to carry over my picture and banner to a new account on a different instance.
  5. I want to edit my account “bio” information for a specific instance.
  6. I want to access custom emoji from my instance. (if time allows)

1. **Public Profile**

As a user,

* 1. ~~I want to be able to follow other users.~~
  2. ~~I want to see my list of followers.~~
  3. I want to change my display name.
  4. I want to be able to pin any of my “toots” to the top of my profile.
  5. I want to mute/block other users.
  6. I want to change my notification settings.

1. **Feed / “List” Customization**

As a user,

* 1. I want to create new “Lists”.
  2. I want to edit and delete “Lists”.
  3. I want to be able to only view my “Lists”.
  4. I want to enable media only mode for the local and federated timelines. (if time allows)
  5. I want to change my filter for words/phrases. (if time allows)

1. **Notification Customization**

As a user,

* 1. I want to change the frequency push notifications are generated.
  2. I want to choose what types of notifications I receive (favorites, boosts, mentions, new followers).

1. **Client Appearance**

As a user,

* 1. I want to customize the primary and secondary color schemes of the app’s user interface. (if time allows)
  2. I want the app to react properly to different font sizes. (if time allows)

1. **Server/Instance Control**

As a server admin,

* 1. I want to be able to ban accounts. (if time allows)
  2. I want to modify the instance’s default primary and secondary color schemes. (if time allows)
  3. I want to be able to read user-submitted reports. (if time allows)

1. **Security**

As a developer,

* 1. I want to implement a safe method of passing over login credentials to the instance without risking the personal information of users.
  2. I want to avoid storing raw personal login information.

Non-Functional Requirements

1. **Architecture and Design**
   1. The front end will be developed in Android Studio.
   2. Java API for Mastodon will be used for server-client communication.
   3. Multiple versions of Android will be supported.
   4. Application size will be less than 35 MB.
   5. Cache size of media, user lists, etc. will be less than 100 MB (if time allows)
   6. The client should load the instance with low latency as a priority of usability.
2. **Usability**
   1. The interface will be intuitive, sleek, and easy to navigate.
   2. Swipes and other gesture-related controls for touch screens will be implemented.
   3. The interface will adapt to multiple screen sizes. (if time allows)
   4. Portrait and landscape screen orientations will be supported. (if time allows)
3. **Security**
   1. We will build user trust and confidence by storing information in a secure manner and only if needed.
   2. Local storage backup of account information will be opt-in. (if time allows)
   3. We will limit/completely avoid handling of sensitive information by any third party or external service. (save Mastodon)