CONNEC+

Sprint 1 Retrospective*

Connec+ (ConnectPlus)

Team 9

Cameron Glass, Calvin Henry, Victoria Oldson, Nirali Rai, and Joey Vinyard

*Please note that this retrospective was done referring to our original Sprint 1 Planning Document.

An updated Sprint 1 Planning Document has since been created.

What Went Well

User Story 1

- **Set up backend server**
  - Used Node.js to build a basic http server that can service GET/POST/PUT/DELETE requests from the client, as well as communicate with the database.
- **Set up Firebase**
  - Set up the Firebase bucket, along with the AngularFire2 services in the client to allow communication with our database and authentication servers.
- **Create base Angular App**
  - Used the Angular command line tool to generate a base app to be built upon.
- **Implement web client navigation and routing**
  - We used Angular routing to allow the page to be switched, without reloading the entire view. We also implemented guarded routes, so that only authenticated users can view certain pages.
- **Implement client-side services to send login information to the backend**
  - We created Angular services that the login and sign up pages call on to access the Firebase API and log the user in.
- **Create authentication pages**
  - We created nearly identical pages where users can signup or login to the service. The login page takes a username and password and includes a "forgot password" routing button, while the signup page takes a username and password.
- **Implement account creation with Firebase**
  - Used Firebase utilities in AngularFire2 to communicate with our authentication server and create account.
- **Implement capability to login and maintain session in web client**
  - Similar to creating an account, we used Firebase utilities in AngularFire2 to login, then we maintained our session in the local cache of the web browser.

User Story 2

- **Create a reset password page on the web client.**
  - We created a page where user's input their email to reset their password that is accessible via the login page.
- **Implement a service to communicate with Firebase to send password reset email.**

- We used a a built in utility in Firebase to send an email to the user, requesting the password to be reset.

User Story 3

- **Create a profile creation page**
  - We created a page where user's input information for their profile that is accessible via the signup page.
- **Implement services in the web client to send new profile information to the backend**
  - We took the user's input from the forms on the front end and stored them in a model in the services. This was then converted to a JSON object to be sent to the backend.
- **Implement backend utilities to formalize the data received from the client, and send it to the database.**
  - Took the JSON object received from the client, and separated it into different categories of information, and stored it in Firebase.

User Story 5

- **Create Settings page**
  - We created a settings page where users will be able to update their invisibility, general information like age, name, gender, security information like email and password, and interests. They are also able to delete their account from this page with proper confirmation.

User Story 6

- **Create a Map page on the web client.**
  - We created the map view page which features the large interactive map at the top. This utilizes the Google Maps API to improve functionality simply. Below the map, users can find their current profile image, name, and mood status. To the right, they can find a toggle button to go to the list view, and a filter button that will help them filter their results. At the bottom is a button to route the user to the settings page to change their information.

User Story 7

- **Create the List view page.**
  - We created the list view page which features the large interactive list at the top. Below the map, users can find their current profile image, name, and mood status. To the right, they can find a toggle button to go to the map view, and a filter button that will help them filter their results. At the bottom is a button to route the user to the settings page

to change their information.

User Story 8

- **Create the change account feature on the settings page**
  - We created a section on the settings page where users could change their account email address and password
- **Implement changing password service**
  - We took the a new password from the "Settings" page, and changed the password for that user.
- **Implement changing email service**
  - We took a new email from the "Settings" page, and changed the email address for that user, and sent an email saying that the email was changed to the old email.
- **Test that features change database**
  - Verified that in the authentication server that emails were properly updated, so long as a valid email address was supplied.

User Story 9

- **Add delete account button on settings page**
  - We created a section where a user could delete their account on the settings page.
- **Implement a service in the web client to delete an account**
  - We took the current password from the Settings page, and deleted the user account using the built-in Firebase method

User Story 11

- **Implement translations for most hard coded text to allow an easy transition in language.**
  - We used a Google Translate service to translate all of the static text in each of our views.
- **Implement a setting that allows the user to select a language.**
  - We added a dropdown menu in the splash bar where users can choose whichever language they prefer

User Story 12

- **Implement the ability for the user to enter feedback in the settings page in the web client.**
  - A feedback form was included in the settings page that would make a POST request to the backend.

What Did Not Go Well

For this sprint, we took on a bit more than we could do. While we did a significant amount of work, we made our user stories too large and were not able to complete many tasks. Specifically, much of the backend work was not completed, including connecting social media accounts to the user's Connec+ account and location services.

User Story 3

- ➔ **Design schemas in Firebase for storing user information.**
    - ◆ Whilst we made schemas for our users, we did not get far enough along to fully complete the other schemas for our database.

User Story 4

This was an example of a user story where we definitely needed to break it down into smaller components. For our future sprints, we've dissected this story into 3, one for each api we want to load data from.

- ➔ **Implement logging in to social media accounts through ConnecPlus**
    - ◆ We failed to login through any social media other than Facebook.
- ➔ **Implement retrieving data (Friends/Followers) and storing it in the database.**
    - ◆ We failed to request any data for any of the social media accounts other than Facebook. In addition, we discovered that Instagram's API has been changed so we no longer will be able to request the information that we wanted to retrieve from that source.
- ➔ **Implement a service to request latest information about the user and display it on their profile.**
    - ◆ This was not implemented by any of our social media's, due to our sprint being too large, and not breaking the social media down into multiple stories.
- ➔ **Implement the ability for the user to remove connected social media.**
    - ◆ This was not implemented by any of our social media's, due to our sprint being too large, and not breaking the social media down into multiple stories.

User Story 5

- ➔ **Implement services to send a change in settings to the backend.**
    - ◆ We had planned to include services to allow the user's preferences and settings to be changed, however, along with the following features, we scheduled too much for this sprint and were unable to add them.
- ➔ **Implement backend services to store settings in the database.**
    - ◆ We had planned to include routes in the Node.js http server to allow the client to send the updated settings to the backend, and for the

server to update them in the database.

- **Implement services to read and store settings in the local browser storage.**
  - To save on bandwidth usage, instead of reading the settings from the database every single time, we instead elected to store them in the local cache of the browser when the user opened the app or update their settings.
- **Implement services to request user settings from the backend if local storage does not contain them.**
  - To help maintain robustness of the application, we planned to check if the settings were already in the local storage, and query the database for them if they were not.

User Story 6

- **Implement ability to obtain current location from the user.**
  - It was planned for the application to determine the location of the user using built in browser capabilities. First we would have to determine what browser the user was using, then implement specific services to request the location from the user.
- **Implement ability to convert current location to geographical coordinates.**
  - We planned to display the locations of the users on a map via geographical coordinates
- **Implement a service to request nearby users from the backend.**
  - We planned to get other users near a specific user's location within a given range
- **Implement a backend service to query the database for nearby users.**
  - We planned to be able to query the database to retrieve those nearby users and send them to the front end.
- **Implement a service to parse the user locations and display them on the Map page.**
  - We planned to take the locations of nearby users that we requested and display them on the map.
- **Implement the ability for the user to navigate to user profiles from the map.**
  - We planned to allow users to scroll through the map and look for nearby users

User Story 7

- **Implement ability to parse nearby user information and display it in list format.**
  - It was planned for the client to parse the data of nearby users received from the backend, and display it in our list view. However, these features were not completed due to overscheduling during this sprint.

- **Implement ability for the user to navigate to user profiles from the list.**
    - It was planned for the user to be able to navigate to different users' profiles from the list view by selecting them in the list. This would navigate the user to the profile based on the uid of the selected user.

User Story 9

- **Link service with front end UI**
    - We intended to write code which would take the input from the web page, and send it to our database. We did not end up with enough time to implement this task.
- **Test that the change appears in the database**
    - Since we were unable to complete all tasks that went along with this story, we were also unable to write unit tests to test those tasks that we were not able to complete.

User Story 10

- **Implement front end links from user icons on the map view to their profile**
    - Our goal was to make it so that the icons on the map would link the viewer to the icon's profile providing them with more information about the user. This was not implemented.
- **Implement front end links from user items in list view to their profile**
    - Our goal was to make it so that the icons on the list would link the viewer to the icon's profile providing them with more information about the user. This was not implemented.
- **Query backend for selected user profile**
    - We planned for the Angular component to first parse the url of the user page to find the uid of the user to be displayed, then make a GET request to the backend to get the specific user form the database
- **Display User profile**
    - Our goal was to display the selected user's profile on the original user's screen
- **Test that List view correctly Displays Profile**
    - It was planned to build Angular Unit Tests to query the backend for specific profiles(Valid and invalid), and check if the correct data was displayed.
- **Test that Map view correctly Displays profile**
    - It was planned to build Angular Unit Tests to query the backend for specific profiles (Valid and invalid), and check if the correct data was displayed.

User Story 12

- **Implement the ability for the backend to store the feedback in the database.**

    - It was planned for the backend to have a route that would have the ability to store the feedback it received from the client in the database. However, due to overscheduling and a lack of time, these features were not completed.
- **Implement a service to allow the web client to send the feedback to the backend.**
    - It was planned to implement services in the web client to make a POST request to the backend, passing the feedback on to the server.

How To Improve

- One large issue we had throughout Sprint 1 was the planning of how much work we should be accomplishing in the sprint alone. Wanting to ensure that the project was enough work for the semester, we felt pressured to push a lot of work into the first sprint so that we would have time to branch the project onto other platforms. This caused us to feel overwhelmed and possibly not achieve as much as we could have.
- In Sprint 1, we completed nearly all of our front-end work, however, much of our backend stories were left unfinished. For Sprint 2, by spending more time and energy on backend work, we should be able to get more features completely implemented.