First Sprint Plan Goals

Team 9/BOWSHOT

10/6 - 10/18

B Grade

Set-Up & Meta tasks

- Tomcat
    - All four developers install Tomcat locally.
    - All four set up appropriate roles for Tomcat locally.
    - All four sync Maven and Tomcat.
- Jira
    - Sprint one is created.
    - Create stories for tasks below, estimate difficulty, and assign.
- SRS
    - Flesh out 'definitions' section.
    - Create functional/non-functional distinction.

Code

- Deploy Application Shell
    - Set up Project shell with Tomcat code on AWS equivalent server.
- Enable users with the credentials to enter the shell on a public computer via a browser.
- Create prompts for users who enter the shell to enter basic 'profile' information:
    - Username(name displayed in app), email, first name, last name
- Create Java classes with the eventual goal to enable one-to-one messaging, with a conscious effort to extend for group messaging in a later sprint. Initial classes are:
    - A `User`. A `User` has attributes such as an email address, a first and last name. A `User` has methods like delete account, change username. A `User` can also send `Message`s.
    - A `Message`. A `Message` has attributes such as content, a time/date sent, a sender, and recipient(s).
- Build out a basic user profile, based on the `User` object, which represents users of BOWSHOT who log in to the web application. This will be created upon log in.
    - Use the *builder pattern*. Eventually we will want to create 'government' type users, and moderator type users. Or maybe other types of users. So create the foundation for this now.
- A user can send a message to another user. The receiving user will be able to see the message sent to them on the screen -- this might require a refresh.
    - Sending users will be able to type in a textbox, hit enter, and the message will appear in the window.

Design

- Build Draft UML Diagrams
- Refine UI/UX Wireframes/Diagrams - More realistic UI wireframe drafts

A Grade

- Users will be able to create an account with their email and password.
- Users will be able to ‘log-in’ to an account they created in a previous instance of the application with their email and password.
- A simple system to store usernames and passwords in plaintext -- some sort of mapping database.
- Create a bad sign-in handler.
    - If the user inputs a wrong username/password combination, they know what they did wrong(and potentially their username doesn't disappear if it's valid).
- A user can delete their account. This will delete any objects associated with their `User` object, including message history shared with other users.
- Users have icons
- A user can see all other users on the left side of the screen.
    - They will be able to select a user from the users available to start a message.
- Create a dynamically-updating list of available users, revealing to those who enter the site the other people who have created an account.
    - This list should refresh periodically, automatically.
    - This list should show usernames.
    - This list should accrue people who log in and submit a username.