This document describes the flow of the Snippet Login User Story for scenarios with and without error during the process.

Happy Path

1. User reaches Snippet login page
2. User chooses Facebook login option
    a. User is redirected to Facebook Authentication
    b. User authenticates with Facebook
    c. User is directed back to Snippet
    d. User is authenticated with Snippet and landing page is displayed
    e. End of login sequence, exit
3. User chooses Snippet login option
    a. User enters Snippet credentials
    b. User is authenticated with Snippet and landing page is displayed
    c. End of login sequence, exit

Error States

User cannot reach login page

1. Display 404 page

Facebook auth server could not be reached

1. Return error to user, suggest trying again
2. If Snippet credentials exist, try logging in via those instead

Facebook authentication fails

1. Return error to user, suggest trying again
2. If Snippet credentials exist, try logging in via those instead

No Snippet account connected to Facebook

1. Return error, suggest checking if Facebook account is correct and show Facebook login button again, Snippet login form, and a “Create Account” option in the event they have never used Snippet but thought they had and tried to authenticate

Snippet credentials are incorrect

1. Return error, suggest trying again, offer password reset / username lookup by email

SnippetDB could not be reached

1. Return error, suggest trying again later

Framework Decision

For our project, we decided that the MEAN stack would best suit our needs for a scalable, robust and simple framework. Since it is ideal for building dynamic web applications using angular and uses javascript throughout the stack, it would allow us to create our one-page application with the minimum functionality we require with greater efficiency than learning multiple languages and integrating them together. Angular also allows us to follow a MVC architecture, which would enable us to push updates to the users frequently. Node.js and MongoDB are both very scalable languages which for application is crucial in order to keep up with possible growth of users and their data.

We chose this stack over alternatives like LAMP mainly due to the greater number of unknowns introduced by the number of different languages that we would have to teach ourselves. Since time is of the essence, we need to ensure we can all help each other out and not using one language will greatly impede that. Other disadvantages include the heavier nature of the Apache server framework, the clumsier scalability and slower speed of relational databases like MySQL, and the constraint of having to work in a Linux environment.

The largest challenges we face as a team though is that we're on the whole relatively inexperienced with javascript and dealing with asynchronous I/O and callback hell. However these drawbacks still does not set MEAN far back enough to make it anything but our best choice for this project.