Authors: Ben Corn, Adrian Law, Yingfeng Chen, Namir Fawaz
Date: May 5, 2017
Final Presentation Deliverables

****

Team Meeting Schedule and Completion Dates

****

Date	**Designated Goal**	**Important Notes/Updates**
2/12/17	Each Team Member will present possible ideas for the project app	- 4 Ideas were presented - Music streaming app, scheduling app, event-planner app, food finding app - Top two chosen were Music Streaming and event planner app, top choice being event planner
2/14/17	Finish writing up the pitches for Deliverable 1	Completed
2/15/17	**Deliverable 1 Due**	**Submitted**
2/24/17	Work on Deliverable 2	- Had to pivot to music stream app Snippet as other group did event planner app - Split the app into 4 different states - Assigned a state to each team member to write user stories on
2/25/17	Finish and put together user stories	Completed
2/26/17	**Deliverable 2 Due**	**Submitted**
2/27/17	Brainstorm about the paths for each state in app and how they correspond	- Put together sketch of how things should generally connect - Created similar UML Diagram draft from class
3/1/17	Finish up Deliverable 3	- Only finished up UML Diagram and Happy Path - Need to finish up Analysis part on architecture
3/2/17	Finish up Deliverable 3 remotely	- Decided to use JS for both the front-end and back-end to make it simpler for us - Using mongodb as our database - Finished up the rest of analysis
3/3/17	**Deliverable 3 Due**	**Submitted**
4/1/17	Plan API Prototype	- Decided to do song search bar - Tapped into Spotify API for their

		song database and retrieval - Was able to get music player along with it
4/3/17	Finish Deliverable 4	Completed
4/4/17	**Deliverable 4 Due**	**Submitted**
4/9/17	Plan Database Usage	- Wanted us to make a database for user searches and cache it - Decided that it would not be beneficial to our app - Decided to do login info cache
4/11/17	Finish up login page for D4	Completed
4/13/17	**Deliverable 5 Due**	**Submitted**
4/26/17	Be sure to have most of app completed by now	- Most of the app's core functionalities have been implemented. - UI has also been neatly done. - Facebook Auth still needs work but that should be it.
4/27/17	**Final Presentation**	**Killed it? :)**
4/27 - 5/4	Put Finishing Touches on app and documentations	Completed
5/5/17	**Final Project Deliverables Due**	**Submitted**

****

User Stories Documentation

****

Login / First Time User (BC):
Snippet's login system will be using a variety of third-party authentication methods. It will give the user the option of creating an account or logging in with either Facebook, Spotify, or traditionally with email and password. When first opening the app, if the user is not logged in then he/she will land on the login menu. There will be two scenarios for the login of Snippet.

The first one is if the person is a first time user or the user wants to create a new account, then the user can simply click on the "create new account" button. Snippet will then present the user with one of the various third-party authentication methods listed to create a new account and link them!

The second scenario is if the user is not logged in but he/she already owns an account for Snippet. In order to login, there will be a place for the user to type in their email and password if that is what they used! If they used Facebook or Spotify, then they will be able to click on the respective buttons for login and enter their credentials. Once authenticated, the user will be brought to their profile landing page!

that is what they used! If they used Facebook or Spotify, then they will be able to click on the respective buttons for login and enter their credentials. Once authenticated, the user will be brought to their profile landing page!

Adding a Snippet (NF):
A logged in user will initially land on his or her profile page. To add a snippet to their stream, the user will click on the minimized window on the right side of the screen. Upon clicking this window, their current profile window will minimize as the next page expands. Once the page has opened, the user will be presented with a search bar that can be used to search for songs. Once the user has searched for and found the song they desire to post on their stream, they will be presented with a button allowing them to add the song onto their stream. When the song has been added onto the user's stream, the button will not longer be accessible, and it will notify the user that the song has already been posted to their stream. The user can then move back to their profile page by clicking on the minimized window on the left of their screen, expanding their profile while minimizing the current page. At this point, the user will be able to see their stream updated with the song they have just added. The snippet will automatically be removed from the user's stream after 24 hours.

Adding Friends (BC):
Users are able to authenticate with Facebook to find friends who are also using Snippet - this is

Adding Friends (BC):

Users are able to authenticate with Facebook to find friends who are also using Snippet - this is a crucial aspect of the application to ensure users are able to connect with their social circles. In addition to Facebook, users are able to add friends directly via email / username.

A logged in user will expand the container for adding friends and be presented with an authentication option for Facebook if the user has not already done so. A list of friends who are also using Snippet will be displayed and the user can click the "Add Friend" button if they wish to add them to their circle on Snippet. A "Search by Username" button will also be visible, allowing users to enter the username of a friend to directly add them. Once the user is found, the "Add Friend" button will appear.

If no friends are found from Facebook, no results will be returned and the user still has the ability to search for users directly by username.

Looking at Friend's Streams(AL):

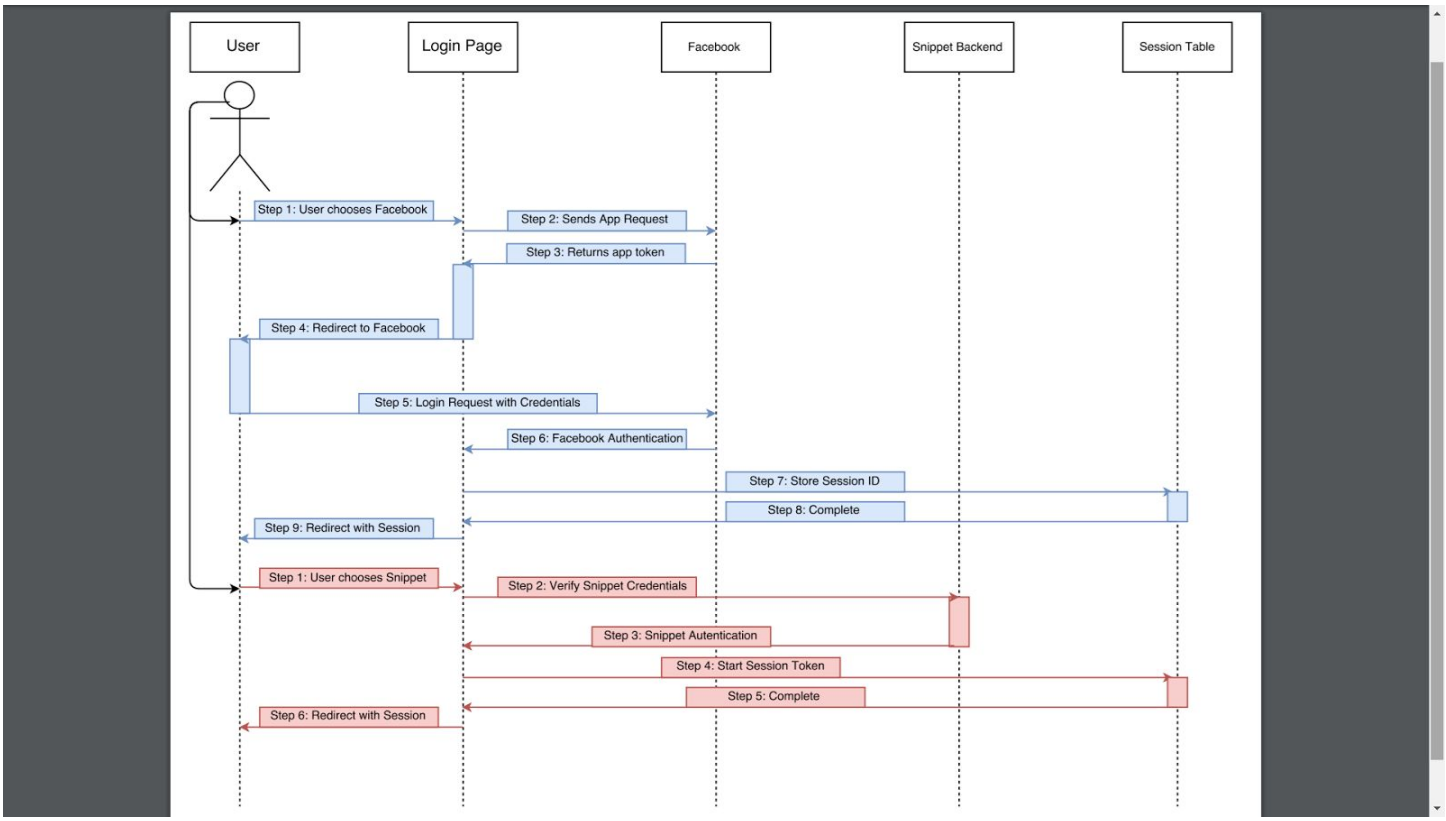
Users will be able to view other friends streams that they have added to their snippet profile. From the initial landing page, a user will be able to see that the rightmost tab of the app is labeled Friend's Streams. After mousing over and clicking on the tab, it will expand to cover the main portion of the screen and populate the list of their friends. If the user has not added any, then it will only display a message saying so. The list will initially be ordered alphabetically by first name or username in descending order, but can be reordered by time updated as well and be in ascending or descending order. Each item on the list displays the user's profile picture, the user's first and last name or username, and a message which indicates how many snippets are on that user's screen as well as now many were not yet viewed by the user. Upon selection of any of the tabs in the list, the tab will expand horizontally down to display all the songs in the user's stream from most to least recently added. It will automatically start playing the first song that the user has not listen to and continue downwards till it reaches a song that the user has played or listened to before. At any point the user can pause, play or skip through the songs using the controls in the player, or select a song by double clicking on the song's tab in the secondary list. To close the friend's tab, there will be an x on the upper right corner, or the user can select into any other friend as well.

Revisions/Thoughts/Notes: This was one of the first deliverables that the entire team had to do. We had met together in order to discuss the four main states of the Snippet app. It was rather easy to put all the user stories together. We had drafted out how each user story will be able to connect to each other's. In turn, by the end of this deliverable, we could actually imagine a framework of some type for our work's structure and how everything would piece together. While making the app, we basically followed this user story outline as our page out lines for the actual app. In the first user story, we said that a user would be able to login with spotify. We did not have time to implement this, but we did implement a facebook authentication. Adding snippets to the user's page was implemented almost exactly how it was described in this user story. Instead of minimizing the different windows in the app however, we implemented a system of three pages inside of one window. When the user presses the arrow keys they are able to move through the multiple pages of the webapp. We were unable to implement an adding friends function through facebook, but this is a goal we believe we can achieve as we continue to work on the project. Finally, viewing friend streams was implemented in essentially the exact way in which we described it.

****

UML Diagrams Documentation

****



Revisions/Thoughts/Notes: We created this UML Diagram based on the ones we saw in lecture/class. It was fairly simple for us as we only needed to do Facebook and regular Snippet login. In the process of making this, we actually tried to use more than just facebook -- opening it up to other media handles like twitter and spotify authentication. But after creating part of it and realizing that it would be excessive, we scrapped it all down and decided that Facebook would be the main third party authentication. It would allow us to gather people's friends so that the app can easily access those friends and give users the option to follow them on Snippet.

****

Happy Path and Error Path Documentation

****

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.

Revisions/Thoughts/Notes: Writing out the happy path for the user login/registration state was rather easy. It was very straightforward with obvious errors here and there. They were all met during the process of making the app and it was fairly simple to make sure they were met (FB Auth had some hiccups but they are done now). What we didn't really think about until we had to start moving on with the app was that there were so many more aspects that we had to account for and have paths into. Like after login page to the landing page, and then landing page possibly to the other pages on the app, and what happens when they click things on each page! Most of the analysis that we wrote for the framework decision was pretty much followed throughout the entire process. Although it was at first hard hitting API's and using that to create functionality within our app, it became rather simple and repetitive once we got into the flow.

****

API Prototyping / Database Documentation

****

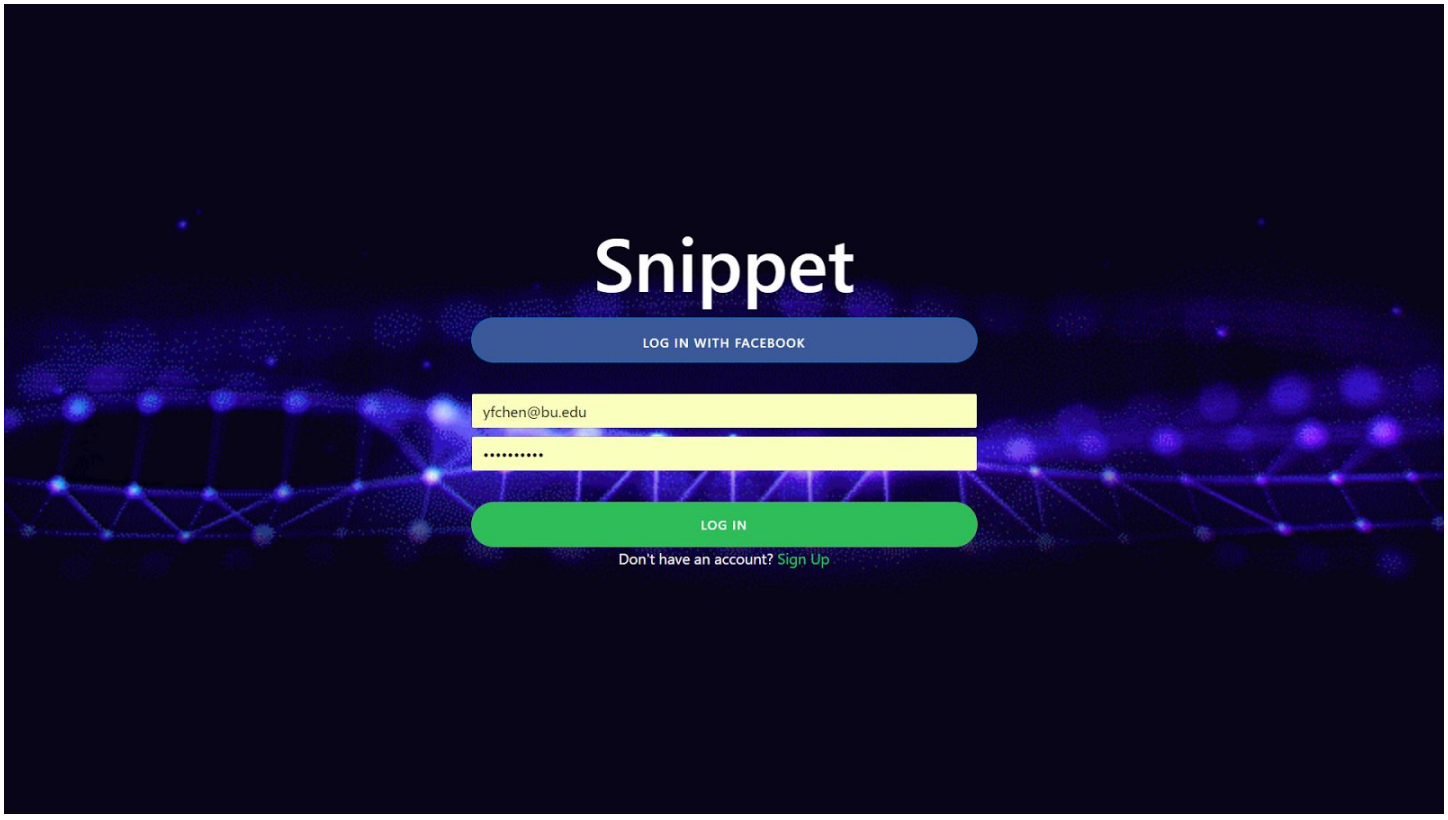
Thoughts/Notes: At first we showcased our app's ability to tap into API's by doing a simple call to Spotify's music search. We thought it would be quite hard to get everything we need in terms of the core aspects of the app. But upon looking at what we could do especially with the API calls, Spotify offered almost everything we needed. We were able to easily get 30 second snippets, download it, add friends, add streams, everything! Another assignment we recently did was the database presentation/showcase. The goals of this presentation was for us to show how we can use a database/cache to store user's searches and history. After thinking as a group about how a cache would be useful for us, we ultimately decided that our app would not benefit much from having a huge database that would store everyone's song searches. The reason being that everyone will make tons of searches -- but the only things important enough to really save were the songs that users decided to post on their stream. So for us to have a database that would store all users searches for some amount of time would really create clutter and slow down the app unnecessarily. We decided that the only time a cache of some sort would really be needed was for the user registration and login function. We would store all login information about the user so that the user, after signing up, will be able to easily reuse their information to access the app.

****

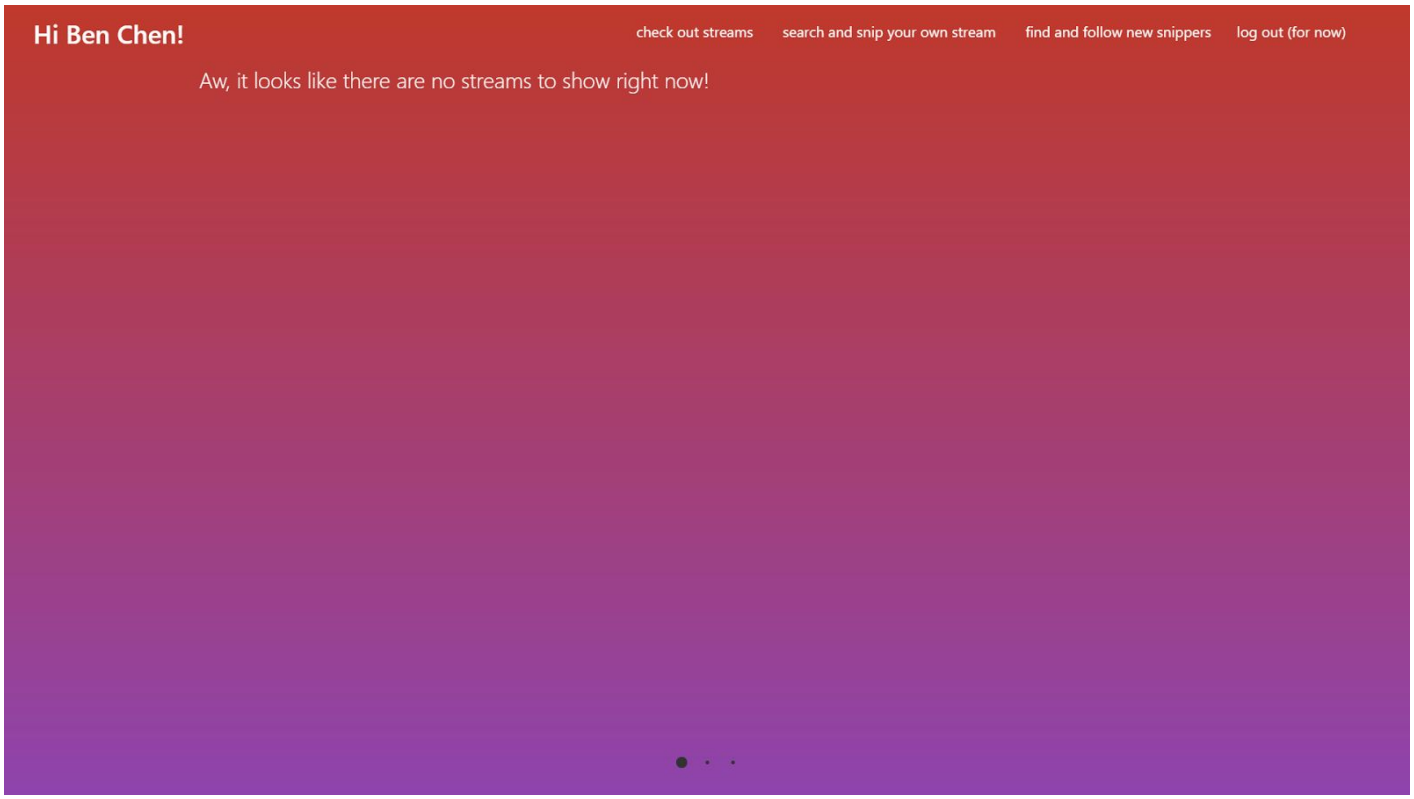
Our App's Final Presentation w/ Finishing Touches

****

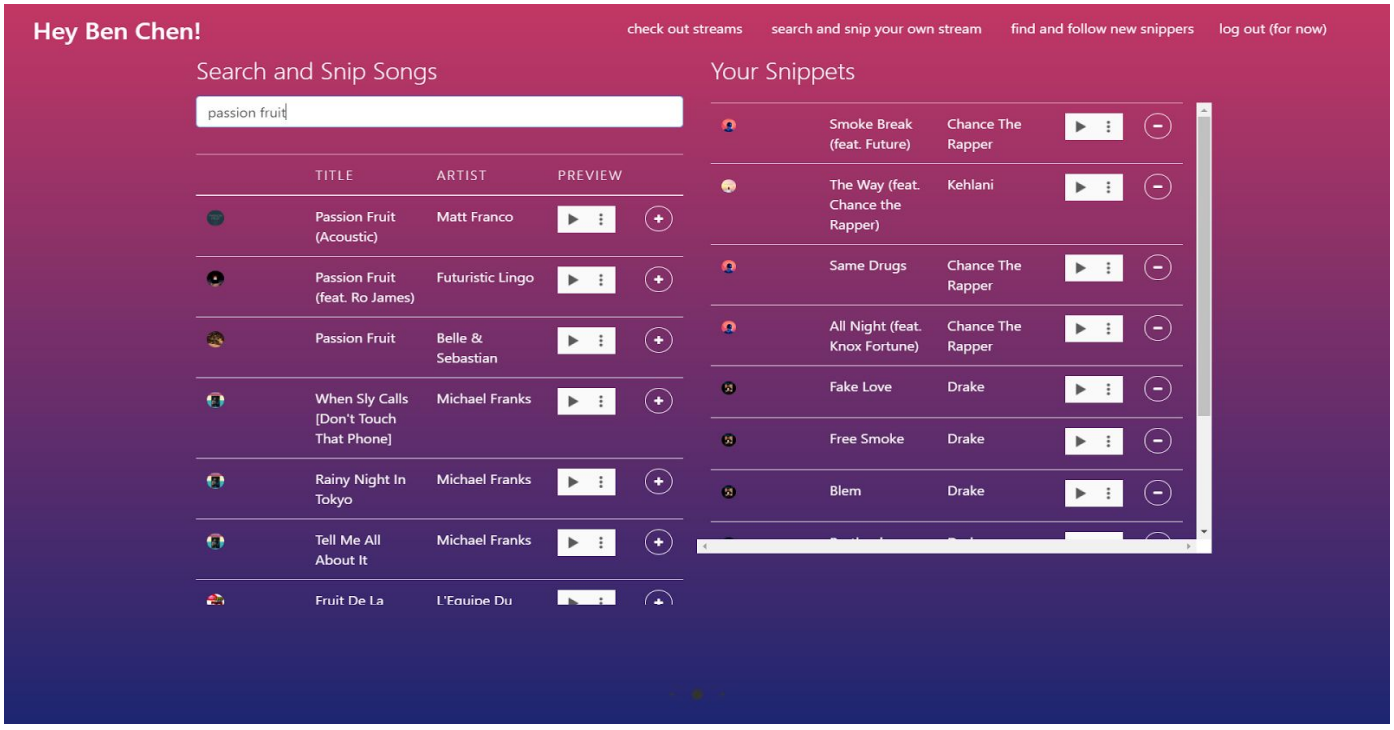
!Important: We have added a few new features and done some improved UI designs since our "final presentation" last week. Here are some screenshots of the changes so that you can see our changes!



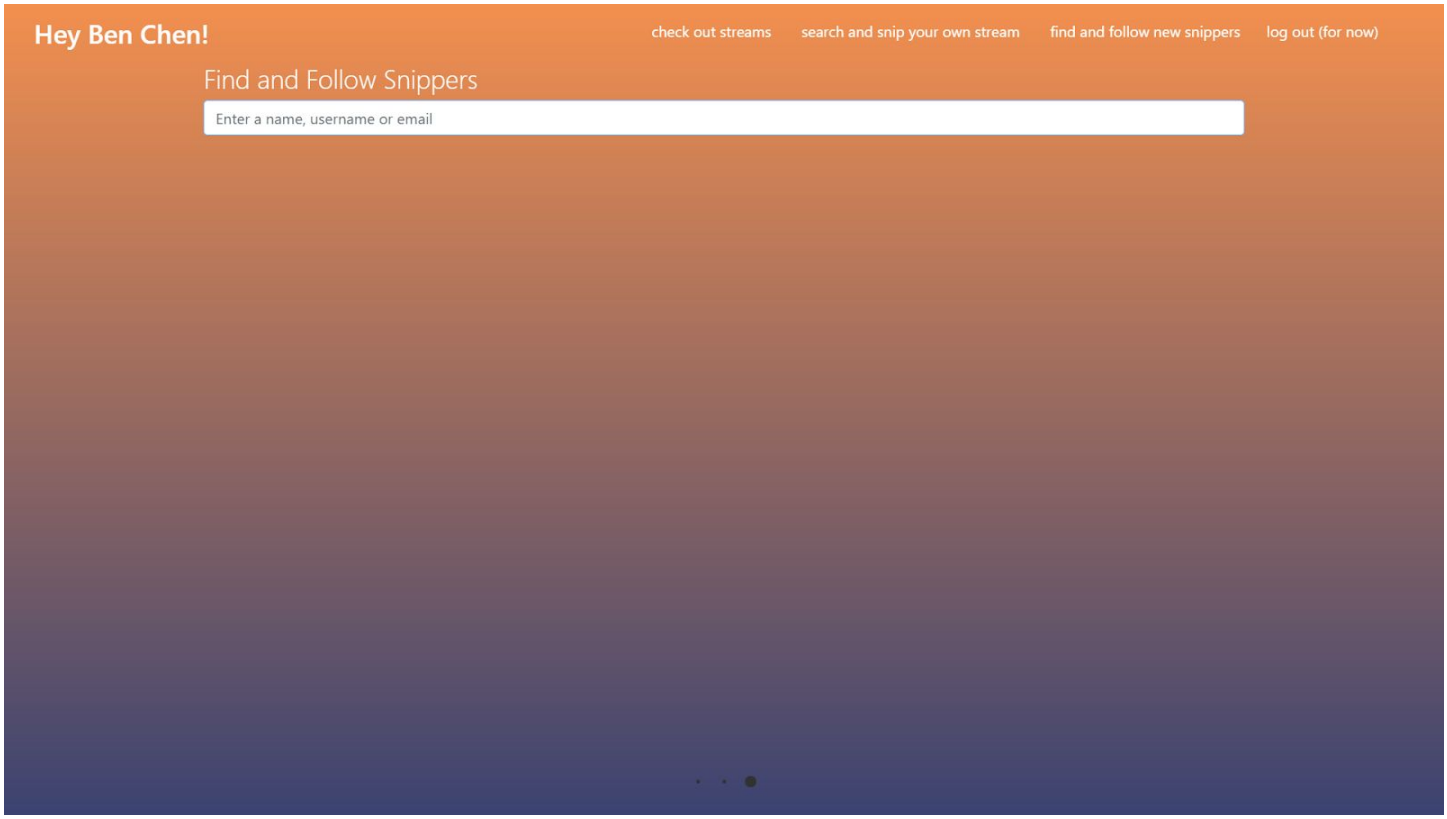
Note for above: This is our new and improved login/sign up page. The background is a gif that actually moves (very nice). The facebook login functionality is now fully functional, allowing us to login through facebook authentication. The regular login and sign up work as before.



Note for above: This is our improved landing page. We now have a simple nav bar on top of every page that helps users navigate between the three known pages: friend streams, search and personal stream, and find/follow people. We have also implemented a logout button now!



Note for above: The UI Design for this page has changed a lot since our presentation. The search bar no longer stretches across the entire page. Instead, it only takes up half the page now and the song placements that it pulls up are adjusted accordingly to that as seen. Your personal stream now takes the other half of the screen, making it seem more full and visible, and nothing is cut off like before. Also, we made changes to the code so now the songs are added and removed from your streams almost instantaneously unlike before.



Note for above: The last page, find and follow snippers, looks exactly the same as it did during presentation. But the functionality of the page has been greatly improved. Before, you had to search for the exact name or email that the registered Snipper had used to login in order to find the person. Now that facebook authentication has been implemented correctly, we are able to just type in a simple name like “Ben” in order to find people.