Snik-Snak Final Report

Project Overview

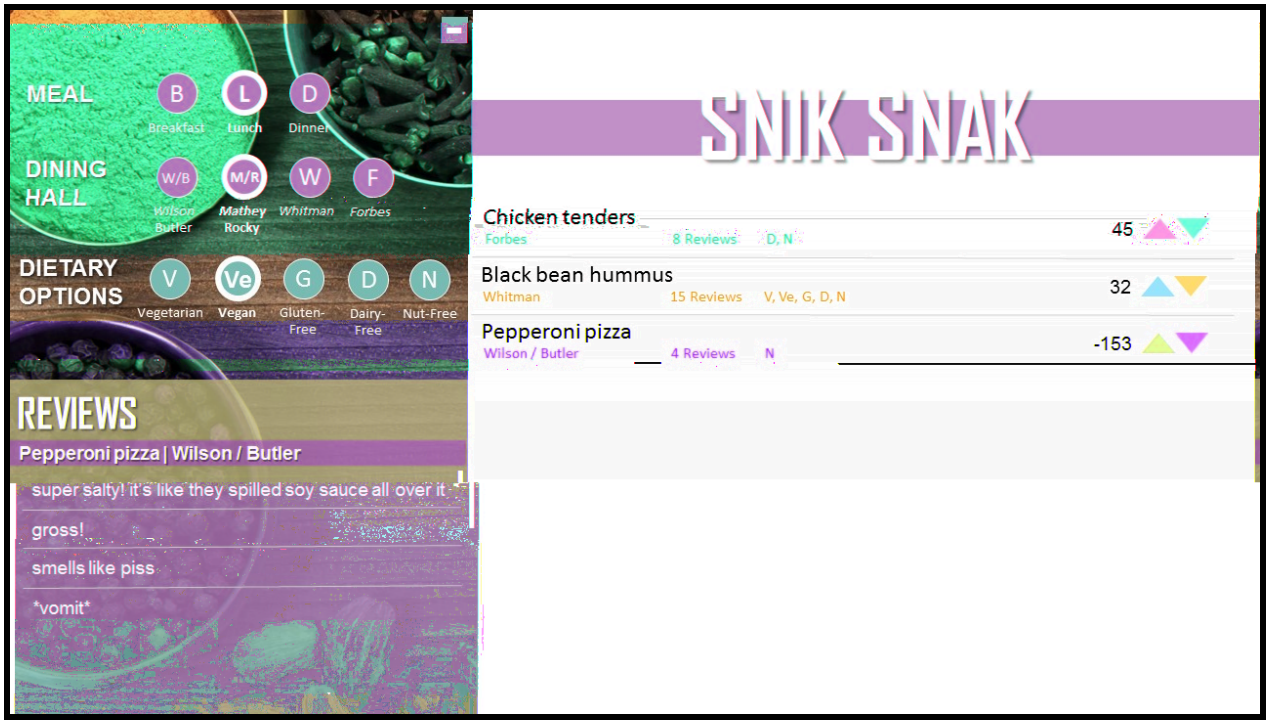
We're proud of our final product as we accomplished our main goal: to make a product that was easy to use and generated new, informative data. Sink-Snak is remarkably easy to use and uncluttered with various features. We did not implement the comments feature which we had planned to in the design document as we decided that our one page layout was much more useful than including a detail view with comments and nutritional information.

Milestone Analysis

When we set our milestones in the beginning of the project, we were purposefully extremely ambitious in order to motivate ourselves to get an early start. We quickly discovered that these milestone were not reasonable, so we spread out our timeline slightly. Because the original timeline was not feasible, we set new goals for ourselves that we were able to meet. One problem we ran into with our milestones was that we set goals for the whole team as opposed to for individuals. We quickly realized that this meant some team members would have had no work and one team member would have had to work unreasonably hard to meet a goal. It was also difficult to set up milestones for individual members as there was often a task that someone else needed to do to allow another team member to finish their part. From this we learned that in order for setting milestones to be effective, we should decide milestones after splitting up the team to work on different sections and write up a calendar of milestones for each team member as well as crucial milestones for the entire team and stick very carefully to it. Though the milestone set in the beginning of the project ended up having very little to do with the actual process, the milestones we set near the end of the project when we were more knowledgeable about predicting how much time a task would take were very helpful in the final stages of development.

Website Front End

The main goal in designing the layout of SnikSnak was to make the food items the focus of the user experience. The secondary goal was to make the website responsive on large browser screens as well as mobile devices for easy navigation and accessibility.

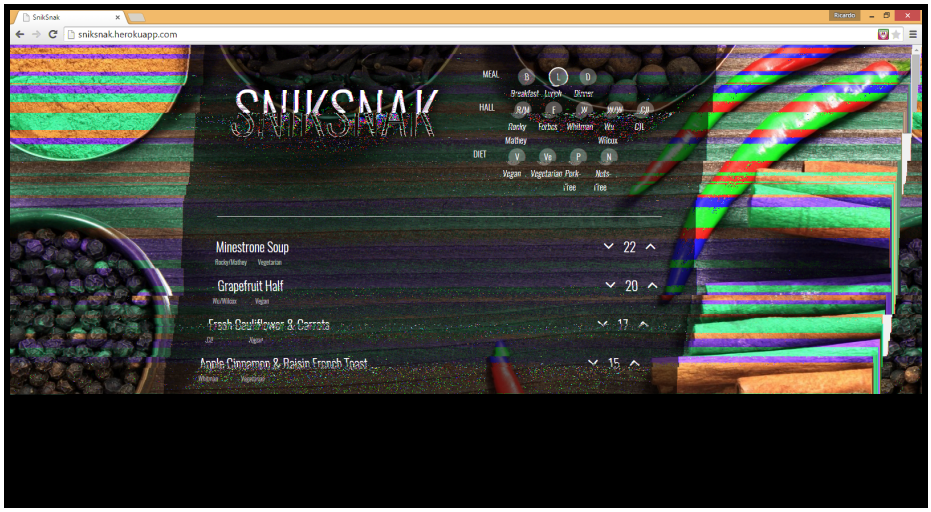


To start out, we experimented with a collapsible window on the left-hand side. In theory, the window would have allowed the vast majority of the space to be dedicated to the food items. It also would have featured a reviews section, where students' comments would appear whenever a food item were selected. With the menu collapsed, the interface would be simple and clean - only a list of the food items.

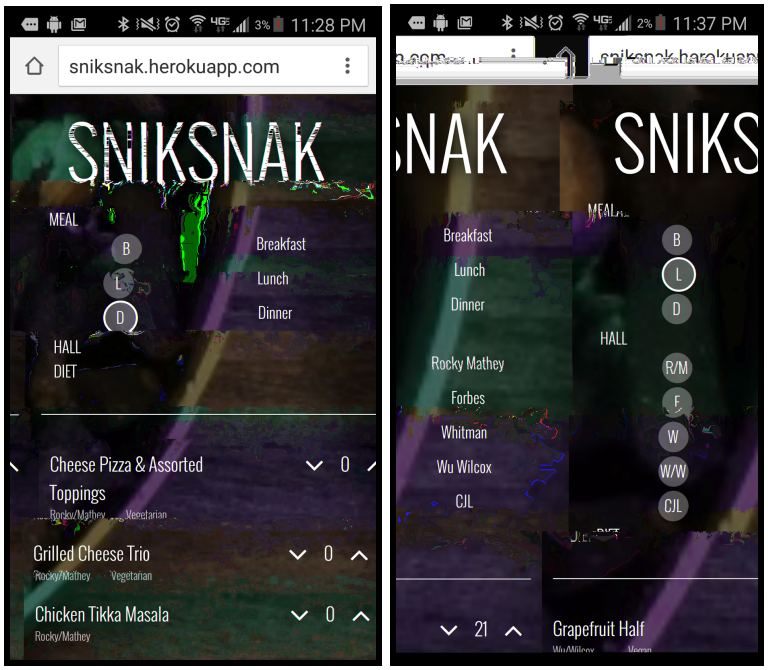
However, there were several problems with the collapsible window. First, we did not know a good way to split a mobile device-sized screen for the collapsible window when it was open; either it took up too much space or it needed more. Second, the collapsible window would have to be open by default because the first thing a user would probably do is set the filters. Thus, closing it would be one more unnecessary click. It was more likely that the user would keep the window open, so we'd have to anticipate what the full user experience would be with the window open. With the window open, the user's eye is not drawn to any particular spot because the filters and reviews section become major distractions from the food items.

In our second iteration, we attempted to create a much simpler layout with fewer moving parts. By implication, we wanted SnikSnak's website to adapt better to small browser screens. The final product uses essentially the above layout for medium-sized browser screens (i.e. smaller than full-screen window sizes on laptops, but large than mobile device screens).

However, we found that on large browser screens, there was too much wasted space, especially at the top half of the screen. This meant that fewer food items could be listed at a time. The logo and filters, not the food items, seemed like the main attraction. We wanted to make more efficient use of the space by vertically compressing the top half of the page.



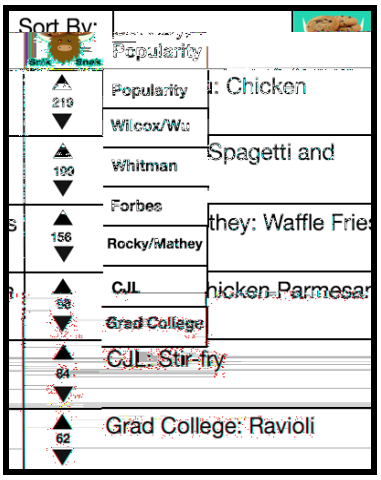
In the final design we have incorporated different layouts for different screen sizes. For a large screen, as shown above, we put the logo and filters side-by-side. We also changed the color scheme to be simpler and more neutral (while still incorporating Princeton orange!). The translucent screen allows us to make smaller side margins, while fully utilizing our background image to evoke a fun culinary experience. These changes maximize space dedicated to the food items, but also keeps filters easily accessible.

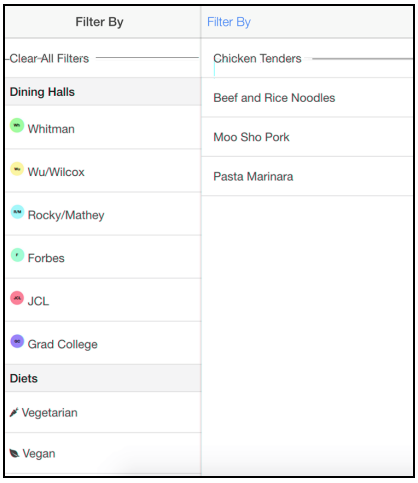


On mobile devices, we switched the structure to organize the filters vertically. Thus, the website uses an accordion dropdown that allows the user to collapse and expand the filters so that the users can quickly hide the filters away when the filters are not in use and focus on the food items.

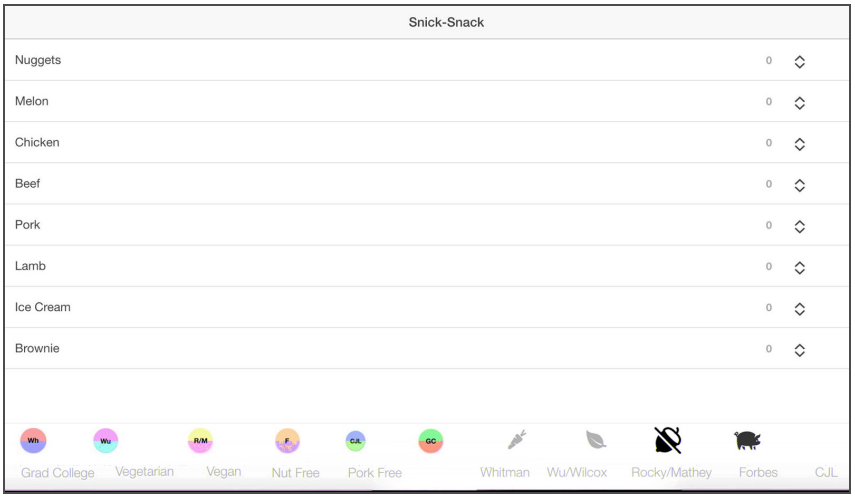
We tested all of our designs with a website at hat would allow us to view the site at multiple different screen sizes, devices, and browsers to check that it was fully functional on each combination.

Mobile Application Front End

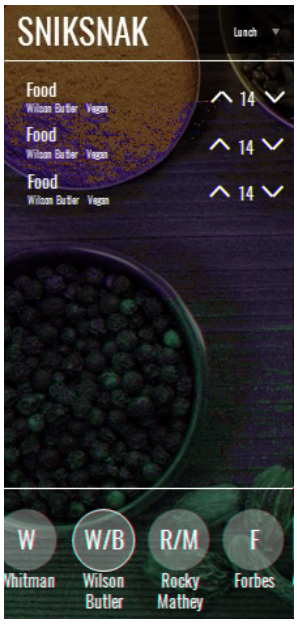


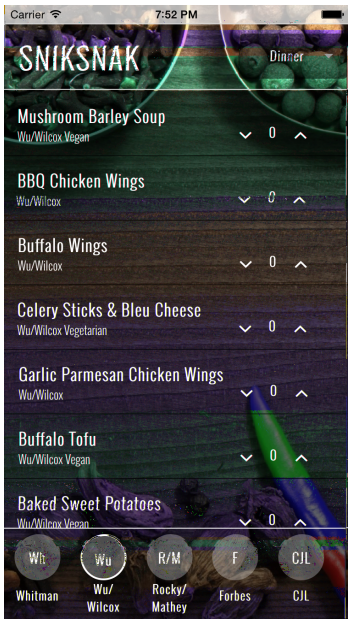


The image on the left is our first mock-up of the mobile application, and the image on the right is the first stage in our development of the app. In our first iteration of the mobile application, we implemented a pull out side menu that the user opens to select their filters and then can close to view the menu items in full size. We decided to change this design because changing filters required an extra two clicks in order to open and close the menu. Additionally, the filters were not always open on the screen, requiring the user to open the side menu to see what filters they had selected. We decided that using some of the space was a better alternative in order to reduce the number of clicks required. This decision was made because our main goal in this project was to make the user interface very simple, as the current system in place is highly complicated. This led to the second iteration:



Here we decided to move the side menu to the bottom of the screen so that you could simultaneously change the filters and see the food of the day. We also formatted the buttons so that the icon is placed at the top of the button and the name of the filter is located at the bottom of the buttons. Though you cannot easily tell in this picture (it's of the application being tested in browser through Ionic), the filters are scrollable so you can easy access all filters.





In this final iteration, our main focus was on making the web and the mobile applications consistent so that users can easily switch from one to the other. We therefore created a new mock-up, as seen on the left, and modelled our application after it (on the right). We added a thin highlight to selected filters, a dark background image with white text, and a meal filter in the top right corner, which defaults to the next meal aiming to reduce the amount of time needed before the user gets the information s/he wants.

Surprises

We quickly ran into a surprise when implementing the REST api. Though the response was displayed on the browser, any GET requests made by our mobile application failed. Looking online for help, we eventually realized that we needed to handle cross-origin resource sharing (CORS). This was simply solved by adding a header to the HTTP packet we sent out, but nonetheless had us stumped for a day or two.

Another interesting surprise was with the time zone differential between our deployed server and our testing machines. When we tested locally, the scrapper correctly scraped food at midnight. However, once we deployed it, we noticed that between 8pm and midnight, no food

was displayed. We then discovered that this was because the server changed date at 8pm EST or midnight UTC. This was a quick fix by setting the timezone to always be eastern time.

Goods and Bads?

One of the choices we made early on that we were pleasantly surprised by was using the Ionic framework for the mobile application. The Ionic framework, especially the Android integration, is still in its developmental stages, so when we started this project, we expected to have some issues with the framework. In the worst case scenario, we were prepared to scrap the entire application and start over with two individual native apps for iOS and Android. However, Ionic ended up working wonderfully for us. We had no trouble converting the code into native apps, and Ionic View, an app that allows displays the app on a mobile device, in addition to viewing the app in localhost on a browser made it very easy to test our apps on multiple mobile devices.

Using Go had its share of pros and cons. While it resulted in clean and readable code, we also lost the ability to set up our database in a single line of code, as some frameworks, like Django, allow you to do. Similarly, Go is a relatively young language with libraries still in development. We had to use many external packages and libraries that weren't fully formed. One particular limitation was GORP, a library that allows us to map structs to Postgres. Though it was our only option for working with Postgres (other than rolling our own library), it did not support arrays as a cell types. This forced us to find a workaround for storing filters of food items.

Doing thing differently

As this was our first time working on designing a functional app, the process of developing the front end was inefficient. We wrote and rewrote code multiple times as we pivoted designs and learned more efficient ways of doing things, sometimes scraping large pieces of code completely. The biggest thing we learned from this is that when working as a team, expectations and ideas in terms of design should be expressed and addressed early on the project, before large amounts of code are written. As mock-ups can be made with little effort in comparison to real code, in the future, we will make mock-ups of the application that everyone can revises and add to in the beginning of the project. We started out with a mock-up that most of the team had not agreed upon, but decided to start building. In the end, we had to redo most of the design to fit with features and details that everyone agreed on. Therefore starting with a new design with input from the whole team in the beginning would dramatically improve the quality of the final product as it would give us more time to polish it. This lesson can also be applied to design decisions for the server and the web + mobile app, though we didn't run into serious rewrite problems here. In the future, we will invest more time in the beginning to the project to creating a design (both front-end and backend) that we all agree upon so the process will be more efficien

Future Developments

Many students would like the ability to keep track of their diet. Thus, we would like to eventually add a detail view that displays nutritional facts for each food item and further develop a tool to allow students to keep track of their diet and receive recommendations to optimize their macronutrient intake each day. To avoid cluttering the current layout, it is likely that the detail view would pop up on a new tab and that the new tool will be completely separate from SnikSnak. Although it may seem inconvenient for the user to switch between tabs, this might still be more streamlined, easier to use, and less distracting.