

CalcuLateMeal Final Report

Planning and Deadlines

Overall, we did meet all of our expected deadlines. I think the key to success was starting simple, iterating on those basics, and then building add-on features once the foundation was locked down. We really thought deeply about how best to implement search and the cart before proceeding which ensured the most basic features worked for our users. Below is a rough timeline of the weekly goals we set for ourselves along with our project advisor, Lance, that we were able to meet in order to keep the app progressing:

Week 1: March 27th

- Initial design template
- Basic server

Week 2: April 3rd

- Bootstrap template added to pages
- Initial cart design using Javascript

Week 3: April 10th

- Search by category and item
- Input pricing data into database

Week 4: April 17th

- Search by keywords
- Sessions IDs and cart functionality

Week 5: April 24th (Alpha Test)

- Suggested items and popular items
- Splash page and search by time
- Birds-eye view map and logo

Week 6: May 1st (Beta Test)

- Cart redesign finalized
- Combos (in cart and on Specials page)
- Packaged Items
- Debugging

Week 7: May 8th (Demo)

- Sorting by item name and price
- Final design decisions
- Debugging

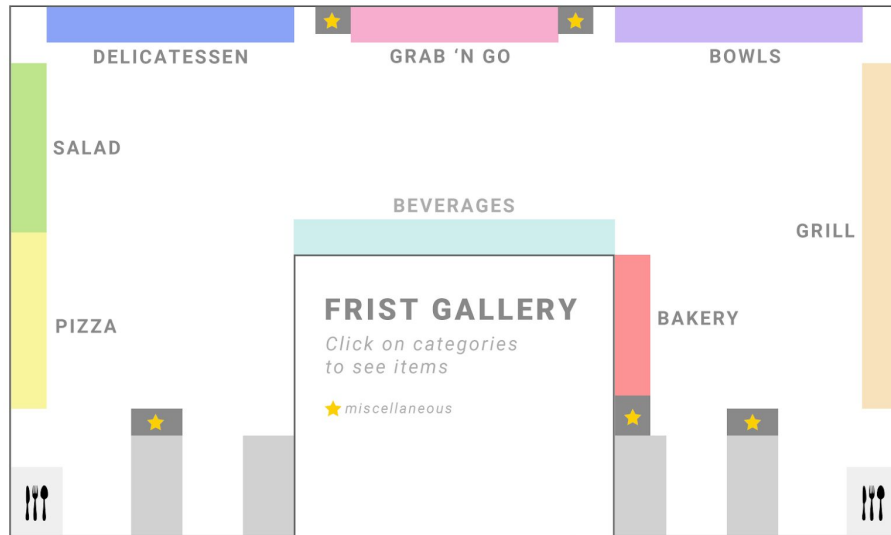
Design Decisions

Looking back at our Design Document now, it seems that our initial planning for the app was thorough and complete, and it represented a well thought-out and feasible plan which we were able to follow closely throughout the semester. Some notable design decisions are detailed below:

Homepage Design

The overall look of the app now is quite similar to the original sketches included in the Design Document, with the exception of the homepage. Instead of having the categories listed with stock images in a grid formation, we decided to create a large map with sections labeled so users can locate late meal items in the gallery through a bird's-eye-view.










One iteration of the homepage design is shown below:



Cart Design

One of the choices we had to make was in the construction of the cart. Originally powered by Javascript in the frontend, the cart was eventually generated through an additional database in the backend. Both of these had costs and benefits. When it came time to add on combos into our project, it became more difficult because we couldn't scrape the items from the cart as easily. Nonetheless, having the cart in the back end did help with maintaining it on each of our pages.

The physical design of the cart was a subject of hot debate amongst our team, and it underwent several iterations before we designed on its final design. At the start, we were debating whether it should be an entirely separate page which the user could click to or if it should appear on each page. We eventually decided on the latter, but then we were torn on how to present various pieces of information. We originally had items in the cart, suggested items, total price

CalculaLateMeal		Home About Favorites	
<div>Search...</div>		<div>Q</div>	
	Guacamole	\$2.00	<div>+</div>
	Hummus	\$2.00	<div>+</div>
	Neapolitan Cheese Pizza	\$3.10	<div>+</div>
	Neapolitan Pizza Deluxe	\$3.95	<div>+</div>
	Pita Chips	\$2.00	<div>+</div>
	Shaker Fries	\$2.35	<div>+</div>
	Sweet Potato Shaker Fries	\$2.35	<div>+</div>
	Toasted Reuben	\$6.55	<div>+</div>
	Tortilla Chips	\$2.00	<div>+</div>
YOUR CART		TOTAL	
Bacon Egg and Cheese \$4.50		\$4.50	
SUGGESTED		\$1.50 remaining	
Kozyschack Pudding \$1.40	Milk Half Pint \$0.95	Hard Boiled Eggs \$0.80	Checkout

(including difference with budget), and the checkout button on the same page and implemented a horizontal scroll for items in the cart as well as those suggested. There used to be lines in between each item as well. However, we soon realized that this design was not the most mobile friendly and ended up becoming very compressed when viewed on one's phone. The scroll also was not as great as we had hoped since a phone's largely vertical stature makes vertical scroll more viable. As such, we modified the cart to include the main information (price and checkout button) at its top and then implemented JQuery so that users could press on the cart and view cart items and suggested items; this facilitated a sleeker design that took up less vertical space. In addition, after hearing user complaints that they didn't know about the touch functionality in which items would pop up, we added a small arrow that changes its direction (up or down) upon each click.

Implementation

We also learned a lot about Sessions in Flask, which allowed us to maintain, preserve, and modify the cart across pages for a single user. Less than two weeks into the project, we had already developed a cart that was working on the front end; however, this cart was simply a mix of HTML/CSS and Javascript and would not stay live across pages - for example, if you added an item from the search page, it would not save that item if you navigated to a different page. When we brought this issue up to Lance, he told us to look into using Flask Sessions. While there were a few technical hurdles in understanding how Sessions worked, it soon became very intuitive to use and integrated well into our application because all user IDs are stored on the client server. Thus, we were capable of creating a cart (a Python list of items) for each session ID and storing this across pages: items were appended to this list when added to the cart by the user and removed from this list when removed from the cart by the user. We loaded strings from the backend containing the cart items into the appropriate html template and displayed the cart as such to the user.

Testing & Implementing User Feedback

CalcuLateMeal was live on Heroku very early in the development process so we were able to get lots of user feedback along the way to take into consideration. Specifically, the front-end design of the application, mainly encoded in HTML, CSS and Bootstrap, was up and running rather early on in the project timeline. Since then, however, the look of the entire site has changed drastically. Designing a user interface requires a constant loop of user feedback and change. The scope of the design adjustments varied greatly; while we made many major decisions - what we decided for the color scheme, how we layed out our index page (transitioning from a scroll of category images to an SVG vector map visualization) and how we displayed the cart on our page were some examples- most of the design tweaks we made were super minute choices, which cumulatively fine tuned the app's appearance to make it the most functional, intuitive, and enjoyable to use as possible. This process generally required a great deal of testing from our users and gathering detail-driven feedback from their experiences. For example, we went up to random people in Frist and showed them different colorings of the main page map and asked them to vote for their favorite. We also gathered user feedback

about the functionality of the app, with many of our close friends actually using it in Late Meal throughout the last 2 weeks. They were able to point out features that were non-intuitive like the clicking of the cart to reveal the rest of it, which led us to add an arrow icon to indicate this specific functionality. Our testers were also able to find small bugs in the program by testing out different combinations of cart items that we had not thought of, ultimately helping make CalcuLateMeal a fully functioning web app.

Surprises & Resolution

We encountered numerous surprises with Campus Dining throughout the project. After a few weeks of trying to get in contact with Campus Dining staff, we received news that they did not have an existing database of item names and prices. Thus, we had to manually collect data from late meal and add them to a CSV file which we then read into our own database. Luckily we were prepared to do so from the start of the project, so this surprise did not have a severe impact on our final project. Additionally, hours after our final presentation, we received an unexpected email from Campus Dining staff apologizing for the lack of communication earlier in the semester and offered the pricing data that they had extracted from the cashier's check out software. This was a pleasant surprise, and we will consider implementing it in the future in order to ensure our menu items are complete and updated.

Additionally, while testing our website, we discovered that searching "cookies" in the search bar always produced an internal server error. This interesting surprise had us stumped for a while. Initially we thought it was related to the keyword "cookies" for browser cookies; we eventually realized that the search result being returned was David's Cookies from our database, but unfortunately the apostrophe wouldn't load from the backend to the HTML template. After surveying our users and random late meal guests, we actually realized that the item name "David's Cookie" is misleading, since David's Cookies are hard to distinguish from regular cookies; thus, we changed the item's name in our database to Cookie (unwrapped).

Another unpleasant surprise we encountered towards the end of our project timeline was the difficult combos feature. Like many of our users, we were quite uninformed on the different Late Lunch and Late Dinner combo deals. It was difficult to implement mainly because there were so many exceptions and rules; for example, two pizzas count as a main combo item, but other main items like burritos and chicken and waffles require only one. Similarly, two unwrapped cookies count as one side, whereas other sides like fruits and sodas only needed one to count to as a side. To resolve these issues, we had to encode complex decision rules to properly detect whether a sequence of items in the cart could be converted into a combo. We also added the Specials page so users have a clearer idea of how to create a Late Lunch or Late Dinner Combo. This was one of the last features we implemented and it was very stressful trying to complete it before the presentations. From this, we learned that we should've started this special feature earlier; we originally expected it to be much easier to code, but given that it is a highly requested feature from our audience and expectations are oftentimes not met, we should've started working on this earlier in the timeline.

Future Directions

When creating the original design for this project, we tried to start with a basic core concept of the calculator, and then we brainstormed additional features that could be added based on complexity and time constraints. Therefore, after achieving the most basic functionality of the web app, there are more and more complicated features that could be added to enhance the app. If we had more time, we would've incorporated the following features:

- **Additional search functionality:** Previously, we had discussed incorporating search by voice recognition and image recognition. Unfortunately, we did not get to these features, but with existing packages and APIs this seems feasible to implement in the future.
- **Expanded audience to drunk meal:** On nights that Prospect Avenue is open, the Frist Gallery is also open late in the night where students can purchase a limited set of items including chicken tenders, fries, pizza, etc. This additional sale of food items, which are not included with the Late Meal plan, is referred to as "Drunk Meal." Adding a menu for Drunk Meal essentially requires an additional set of food items as well as a feature where users can set their own budgets, since this is not included in the Late Meal budget.
- **Customer-based entry for our database:** Since our data is hard-coded in manually and not extracted from a Campus Dining database, it is possible that we may have missed items, or Frist Gallery may have new items or new prices. Thus, we were thinking of implementing a feature where users can input additional items and prices to their carts. This customer-based entry feature will allow shoppers to add missing items to their own carts and will help us identify items we should add to our own database.
- **AJAX Commands:** Currently, the page has to refresh every time an item is added to the cart, which is visually apparently especially on the desktop site. This is because the functions involved with the cart are pretty much all implemented in the python server; ideally we would implement all the cart features through asynchronous Javascript requests. This would make the visual experience on our web app smoother and simplify the backend rerouting that is currently in place.

Lessons Learned

Another aspect that worked really well for our team was the division of labor. People worked on features which they actually enjoyed working on. For instance, Nina handled the brunt of the front-end and design, yet this coincidentally happened to be her strength and passion in our project. While we took turns handling more minor front-end changes, at the end of the day, we still deferred to Nina on bugs we encountered as she was more familiar with them and could sort out the issue much faster than we could. Likewise, for the rest of the team working on the backend and on some full-stack features, responsibilities were clearly split such that we were not dependent on each other for whichever task we were assigned. Arjun, for instance, was working on keeping the cart preserved across pages for each user while Grace worked on the cart design, Kelly on suggested items, and Eileen on the limit of two packaged goods. This

simplified testing as each person knew specifically what part of the project they were working on and could pinpoint exactly what caused the error when their product broke down.