**TigerEats Project Evaluation**

Ishan Sinha, Paulo Frazao, Gabriel Birman, Jamison Mercurio

**Discussion Points:**

* How good was the original planning?
* How did your milestones go?
* What was your experience with design, interfaces, languages, systems, testing, etc.?
* What surprises, pleasant or unpleasant, did you encounter?
* What choices worked well?
* What choices worked badly?
* What would we do with more time?
* How would you do things differently next time?

**Thoughts:**

Ishan:

First of all, I have SO much more respect for web developers now. Going through this project made me feel like a real engineer for the first time at Princeton.

In retrospect, I think the original planning was as solid as we could have made it given all of us had never touched React, Flask, or Heroku before. The timeline we set was slightly ambitious, but realistic, and the stretch goals we set were truly stretch goals. Each team member had a well-defined set of tasks to deliver on, and we maintained a great division of labor throughout. We assigned just one team member to deploy our code to Heroku, which was a fair delegation to make with the knowledge we had, but in retrospect I would have assigned another person to help out. There is so much to learn to deploy on Heroku effectively, and we ended up having large issues with Heroku at the end of the project. Luckily, after lots of sweat, we got it fixed!

Milestone-wise, we did well, meeting almost all of them besides the beta version. One thing I found is that while we thought we implemented our system end-to-end, there were some issues with our Heroku setup that we did not notice until the very end. This is because we did not deploy to Heroku until the last two weeks of classes I believe, and these issues didn’t manifest themselves until I tried to integrate CAS. So, though we did well on our milestones for the most part according to our knowledge of how these technologies should be set up, we had a few deficiencies that crept up on us at the end of the project. So, winter break ended up having more work than expected and we finished things more last minute than we intended, but we got everything done.

My experience with these new technologies was generally positive. I enjoyed learning about the new technologies, and I’m glad we chose React/Flask since that combination is relatively well-documented online. I am a bit disappointed because not a whole lot of what we learned in class ended up being super necessary to understand to carry out the project; the tutorials we had to go through for React and Flask taught us everything we needed. I would suggest teaching React and Flask, as those seem to be the new dominant technologies! Tkinter and Bottle, it seems, are not used in leading tech companies. I did feel like I learned a whole lot about web technologies, which has been enlightening (and frightening) for my daily life!

The only real surprise we encountered is that we were setting up our Heroku incorrectly. We were deploying our whole frontend as a separate process from our backend (which ate up lots of memory and prevented CAS from working -- see our Programmer’s Guide “Design Problems”), instead of deploying a static build of the frontend that was linked to our server. We still had other challenges, but nothing that was completely unexpected like our Heroku issues.

I only worked on the backend and database (though I had to work on Heroku at the end), so my scope is limited. That being said, I am happy with our choices in technology. Everything worked out pretty well. Again, the only issue was our Heroku setup, which we wouldn’t have known in advance.

There are many stylistic changes we would make with more time that I detail in the ProductEval document. The main features I would want to add with more time are: 1) Adjust the Progress page to allow the user to input start and end dates for the time series; 2) Adjust to Student Profile page to allow the user to input a specific date for which to view a student’s meal history; 3) Build an in-application messaging feature with the mobile app -- this would require the mobile app to be fully developed and for our application to draw on the same database as the mobile app, however, which is not feasible for now.

If possible, I would have liked to solicit user feedback earlier. However, we needed to tie up loose ends with the application first.

Jamie:

I think choosing ReactJS and Flask + MongoDB was a good choice in learning different front-end and back-end technologies - and I even had a few internship interviewers say they were impressed I was learning these tools in a university class. Additionally, Heroku was a good practical choice for deployment, and I’m glad I learned that valuable skill that I can apply to later projects when I’m looking to host them at a free tier.

In regard to our original planning, I would say we did well in splitting the team up into front-end and back-end roles. It seemed to work well in having certain team members specialize in different areas, especially as React has a bit of a learning curve. I worked more closely with Paulo on the front-end, and because we both specialized in the React side of the project, we had an easier time answering each others questions and collaborating with similar features. I also think that the goals we set were well spaced out and realistic, given the serious amount of effort all four of us put into this product. However, I do think that bigger aspects like Heroku and CAS should have been done by more than one person each, as it ended up taking much more than that in order to get both of those working fully.

Milestones were as expected - the beginning of the project focused on learning React and Flask, while we were able to hit checkpoints later on like the Alpha, Beta and final project presentation well. I didn’t have any prior experience with building web applications, including coding in JavaScript (or Python before this class), but have found this 333 project to be one of the most practical and useful CS experiences I’ve had so far.

Unpleasant surprises include both Heroku and CAS, which turned out to be a bit harder to set up when working with React and Flask. I had to use a bit of a “hack” that I found online, which, although it worked, made it harder for Ishan to set up the CAS functionality. Nonetheless, we were able to conquer these challenges successfully, and the app now utilizes both to create an intuitive and immersive user experience.

Good choices included splitting the team up by page as well (as that made it easy for us to work independently, such as over winter break), as well as having everyone continually report their progress over our group chat. Moreover, I used some libraries, like MaterialUI for styling and React-Vis for graphs, that made the final implementation of many features much easier, and allowed us to focus on more big picture aspects rather than tediously styling features like buttons and bar graphs. I think it also helped that all four of us became closer friends through this project, and motivated each other to succeed through the late nights and many hours of debugging and testing.

Given more time. I would continually add features to my page like “Meal Notes” (which was a bit of a stretch goal), and potentially make the data more realistic by adding more students and pictures of their meals. Additionally, we could implement other stretch goals like the messaging component, and I think linking the project to an actual iOS/Andriod app would be thrilling (I can’t wait to test it all out when it’s fully functional). Finally, I can’t say I would rather do anything major differently next time, other than splitting up tasks differently regarding Heroku and CAS like I mentioned earlier. It was a great experience with a fantastic team, and I’m very proud of what we accomplished over this semester.

Paulo:

I think our original planning actually worked out quite well. Since we knew that the learning curve for React was steep, we dedicated several weeks at the outset solely to learning it (enabled by Ishan being ahead on some design elements). But that was a great choice, as it enabled us to approach the rest of our more time-sensitive goals with more control over the technology, at least for those individuals who worked on frontend. On that note, the milestones went fairly well too. Honestly, the beta creeped up on us; I pulled a lot of the weight for that deliverable out of happenstance, but we definitely could have planned better for that. Similarly, the presentation came at a very awkward time. We were all busy with finals and Jamie had just left for study abroad, and so that came together well but was not as polished as it could’ve been. The rest of the milestones, though, I think went smoothly and were properly planned out.

Now, in this project I focused almost entirely on frontend development, and I really enjoyed it. React was really challenging, but super cool; the hyper-modularity enforced through components was something I really enjoyed learning, and it was also cool to observe the rendering process and how that all functions. The testing process, which I worked on a bit (mainly in recruiting users), was also kind of interesting. It’s the first time I’ve gotten feedback on such a large-scale project, and it was cool to see all of their opinions come together and be able to study them for trends. Nevertheless, these aspects of the process were definitely positive surprises. At the same time, though, I was not expecting the arc for React coding to be so long. Everything took pretty long to code up in React, primarily at the outset; I think this has to do with how intertwined the components are due to state and props. Nevertheless, that was the only major negative surprise.

In terms of successful choices, I’d say my decision to move forward with Material UI was a great one. This library, which supplies pre-built components, allowed us to spend our time on linking modules together instead of creating them from the ground up, which was an excellent way to save time. Similarly, I really appreciated our decision to divide the work between frontend and backend. This allowed us to understand our specific areas incredibly well, though it was trickier when it came to things like Heroku and CAS. This was our only major mistake, I thought. We pushed Jamie to handle Heroku by himself and Ishan was tasked with working on CAS individually; both of these components proved to be issues all throughout the back half of development, and I feel that that could have been minimized with a bit more planning and delegation at the outset.

I think we would do most things the same way next time, aside for our work split involving Heroku and CAS. As for what we would do with more time, I think we would prioritize implementing a couple more stretch features. The first feature of these would be an in-app messaging feature. The second would most likely be a set of additions to existing pages, like modifications to the progress page and perhaps additional analytics on the dashboard. Overall, though, I feel really good about what we’re managed to put together this semester.

Gabe:

Regarding the system architecture, I think the original planning went well. Our choice to use React for the frontend and Flask with the backend was suitable (with MongoDB as our database). Flask was a good option because of our prior experience with python and bottle, while providing a little more documentation for tasks that we figured we would need guidance for. React was a good option for us because it made for easily moduralizable code, gave us access to extensive external libraries, and documentation regarding its set of features. MongoDB was a clear choice because it’s lack of scheme allowed us to reorganize data quickly to best suit our needs on the frontend. At the outset, we did a good job of conceptualizing the amount of time it would take to develop both the server side and the client side, with the latter taking much more time, so we invested in a paid tutorial series that would help us develop the project more efficiently and effectively.

Though I helped a little with MongoDB setup and some server-side functionality, my main milestone later became developing the student progress page, as it became clear that this was going to be a major task. Because I was switching over so fast from Python to React, which I had merely dabbled in before, I did not have time to truly watch the lengthy tutorial series before meeting the week’s goal of having a semi-functioning beta. As a result, I did not do enough research before choosing a charting library, and chose one that was simple to use and provided similar examples. This made it difficult to switch over to other charting libraries when problems arose with React TimeSeries Charts, as there was a learning curve involved for more powerful visualization tools (e.g. d3.js), while other libraries would not have necessarily provided the capabilities I was looking for due to similar levels of abstraction that prevent the user from accessing certain components.

To give some context for the above paragraph, I’d like to point your attention to the black color of the time series background. Though this might seem like an odd choice as black had heretofore not been detected in the UI color palette, it was necessary in order to both be able to see the label axis and the value axis; the library had not given the users access to the CSS styling of those components, and thus, black was the only choice that would enable the display of the label axis and value axis colors simultaneously. In hindsight, I probably would have forked the open source directory and incorporated changes I needed to the code, but as it currently stands this issue has been addressed with Fix #355 on the Github page

(<https://github.com/esnet/react-timeseries-charts>) which will soon be merged into master.

The difficulty with switching over to another library was compounded by the fact that the initial design I was given was not suitable for a web app, and in fact did not provide the user with clear data visualizations. Thus, I had to leave room for changes when selecting my library, which ruled out more powerful but lower-level tools as they require more time to build components. Though I was given full control over how the page could be formatted, in hindsight I would have taken time earlier to confer with my teammates and user testers in order to properly and definitely mock-up the UI. I did end up making a change from presenting the data as a stacked area chart (as seen in the beta) to the current individual bar graphs after conferring with a user tester, however. The reasoning for this being that it did not make sense to stack data with different units in the same graph, especially when levels of one nutrient would often largely outweigh levels of another, making it difficult to track progress for smaller-valued nutrients. In general though, the abstraction of the library made it easy to use, and the corresponding time series library was similarly simple to use. In the end, I think the page looks quite good and am overall happy with the level of design/functionality the technology I chose afforded me. Ultimately, there had to be a compromise between the amount of functionality and clarity in the app, especially as I made the design choice to prioritize the time series and have it occupy as much of the screen as possible.

I would definitely have liked to have queried user feedback a little earlier, as that would have given myself more time to implement suggested changes. Though the responsiveness of the web app was nice for final usability, it makes implementing changes very difficult as it results in 3x the necessary work to adjust the view in each of the three width ranges and corresponding views. Thus, I would have preferred to have queried feedback with users testing on full width screens and then made the page responsive afterwards.

Some features I would have liked to have implemented with more time: weekly rollups starting on the Monday instead of the Thursday (this was non-trivial as I had to aggregate the data with a weekly rollup being defined as a 7-day rollup beginning from Unix Time -- a Thursday); forking the React TimeSeries Charts library and letting me style the value axis and label axis colors; adding modals to explain how the rollups/checkboxes works (a timerange modal was implemented after user feedback showed this feature was confusing); time range selection using a date picker; mobile web page hover/swipe for tracker line movement.