Team Iron / Judit Chang-Horvath, Michael Colombini, Katelynn Hull, Kira Maximova, Abdullah Pathan, Gavin Rios, John Wasikye

Weekly Development Report #16.

Performance Period: **Wednesday, 04/26 - Wednesday, 05/03 (Final Prototype Demonstration)**

1. **Group Accomplishments:**
    - We built out numerous screens for our prototype.
    - We refined the backend code, and we created several new API endpoints.
    - We presented our final Prototype
    - We finalized the slides for our Final Prototype Demo
    - Finalized Lab 3 and 4

2. **Individual Contributions/Accomplishments:** (to be filled out by each individual)

- **Judit Chang-Horvath**
    - I discussed with the backend team the two main approaches with regard to data being returned to the frontend. Based on the 'Advocacy and Petition Sites' screen, I find it very practical to only return the data that needs to be rendered on the frontend, however, as I learned, this approach is not in line with the best practices of REST APIs. I think that making several API calls from the frontend to the backend, returning a lot more data than actually needed, and then choosing the data points one-by-one that need to be displayed on the frontend is very cumbersome and inefficient. Probably in the future, I will prefer working with GraphQL or some other similar framework.
    - I made the changes to the codebase so that it aligns with the REST API practices, however, it needs further modifications and further API calls need to be made in order to get the Shopping List screen back to the state that it was in a day ago.
    - Dr. Miller's recommended that we only make one API call for each screen. I let the backend team know, and they were able to find a way to make all the data available at the API endpoint that is needed for the Shopping List.
    - Katelynn and I discussed possible ways of implementing the dropdown menus for the LemmeKnow screens (Existing Store, Closed Store) with the new screen setup (using a class and extending the Component class, instead of just using a function - as it was set up earlier).

Katelynn made the necessary changes; she converted the JSON returned by the API call to an array. Now, the dropdown menu is dynamically filled with data from the database.

- I updated the store names, website urls, and email addresses in our database. The ones that we had before were generated by Mockaroo, thus, they were fictional, and I realized that it probably would be better if we would have real store names with real, accessible websites.
- I added the functionality to the backend for determining the recommended container types for specific products. This functionality can be used to retrieve data for the pop-up panel that would need to open once the user taps on the small container icon on the product search listing or on the product entries in the shopping list. Thus, ***we are able to determine the recommended containers for specific products based on product ID.***
- I modified the customized container recommendation screen, and now it dynamically displays the product information. The container information still needs to be determined, and displayed.
- I refined the SQL statement that returns the data needed for the CustContRec screen. The SQL statement returns the number of containers for products with product categories 'dry products' and 'fruits/vegetables'. 'Liquid products' will be an enhancement for later. I informed the backend team, and I asked them to create an API endpoint that makes a JSON object available with the same data points that are returned by the SQL statement that I wrote. Once we would have this work, we could just simply make an API call from the CustConRecScreen.js file, and replace the current placeholders with the keys from the JSON object (the template for displaying the data is already set up). Then, when the page is rendered, the keys would be swapped out by their corresponding values. Thus, ***we are able to determine the number of containers for specific products based on the quantity in the shopping list.***
- I created a store listing screen where currently all the stores in our database are listed because unfortunately, I do not think that we will be able to make Google Maps work. A search functionality based on ZIP code would still need to be implemented.
- I created a store information screen where detailed information (store's name, address, rating, opening hours, website, and email) about individual stores, and a link to the store's products, and a field for rating the store can be found. Not all functionalities work because we haven't found out all the needed steps for connecting the frontend to the backend.
- I created the Profile page and 3 subpages ('Saved stores', 'Share with friends', and 'Profile settings'). Because we do not have a navigation bar, I placed the button for the Profile section into the Home screen. Not all functionalities work.

- I added two sections to the Customized Container Recommendation screen. One section for displaying the total number of containers for doing the shopping at one specific store, and another section for displaying the total number of containers for all the products in the user's shopping list.
- Katelynn and I worked on the Final Demo presentation, and finalized it for our Rehearsal.
- I worked with Gavin finalizing the Product Search screen.
- I created the Search by Icon screen.
- I added a New Search button to the Store Search screen.
- I published the Final Prototype Demo on the project website (both the slides and the downloadable PDF version).
- I worked on the revision of Lab2 Section 1&2, and on Lab3 Version 2.

- **Michael Colombini**
  - [description of the work that was done during the week]

- **Katelynn Hull**
  - I met with Judit to try and get the dropdown menus for the Lemme.Know screens to include the store names from the database. We created an alternate screen in case we couldn't resolve the issue.
  - I figured out how to add an API call to the Lemme.Know Closed Store Screen, made the screen a class rather than a function, converted the JSON object returned by the API call to an array of store names, and now the dropdown menu dynamically gets all the store names from the database. Since the alternate screen didn't need to be used, I removed it.
  - I also updated the Lemme.Know Existing Store Screen's dropdown menu to get store names based on an API call. Then I reformatted the screen to be a class, and reformatted the functions I had originally created to be compatible within the class.
  - I set up the initial slides for the final demo, reformatted some slides, added screenshots, and included new slides based on previous final demos and their contents.
  - Judit and I worked on finalizing the final demo presentation to prepare for the rehearsal.
  - I worked on revising Lab 4 and added red labels to the screenshots.
  - I fixed the day order for the store hours on the store info screen for the prototype.

    - Judit and I finalized Lab 4.

- **Kira Maximova**
    - continue to work on the git Issue items in attempts to get it done for the demo: Nav Menu - run into the challenge of something that Dr. Miller said - try to use only one API per screen
    - worked on website + testing for the demo
    - worked on Labs: 2, 3 & 4
    - updated slides for Final demo (risks, Next steps)

- **Abdullah Pathan**
    - [description of the work that was done during the week]

- **Gavin Rios**
    - [description of the work that was done during the week]

- **John Wasikye**
    - I finalized the container recommendation
    - Created a script for my slides of the demo
    - Practiced the demo and made changes based on the feedback
    - Presented our final demo
    - Tested api calls
    - Worked on labs 3, 4 and 2 group and individual work