Team 11

Purdue Eats

Team Members:

Aniket Agnihotri

Anisha Sinha

Eric Thompson

Mark Jin

Sean Joo

Vaastav Arora

Sprint 2 Retrospective

What went well?

This sprint went fairly well for our team. We completed our user stories and were able to integrate each component such that our functioning, fluid PurdueEats prototype from Sprint 1 was expanded to include more user-oriented meal-logging and settings features. Our user stories for Sprint 2 involved the following: searching for and logging meals, viewing user aggregate nutritional information, viewing menu item-specific nutritional information, entering a user profile picture, managing dining dollars and meal swipes, viewing logged meal history, searching for menu items among others. All of these user stories function as specified in our Sprint 2 Planning Document and provide a good foundation to build on in Sprint 3.

User Story #1

As a user, I would like to be able to see the correct menu items so that I can view what is offered at dining courts, record the right meal preferences, and record my meals.

#	Description	Estimated Time	Owner
2	Create a UI component to render the correct menu items based on the dining court on the menu page.	2 hrs	Anisha
3	Create a UI component to render the correct menu items based on the randomly selected items in the databases for meal preferences.	2 hrs	Anisha
4	Create a UI component to render the correct menu items based on the dining court on the record meal page.	2 hrs	Anisha
5	Manually test rendering of the correct menu items for menu, meal preferences, and record meal page.	1 hr	Anisha

Completed:

Each of the UI components were created for the menu page, meal preferences page, and record meal page in a fashion that matched the UI mockup. The items that were rendered were also correct based on the information that was stored in the backend. As soon as the user was directed to one of those given pages, the menu items were rendered on the screen in an aesthetically pleasing format. The actual functions that did fetch the menu items was difficult to implement and is referenced in the what did not go well section.

User Story #2

As a user, I would like to be able to view a visual representation of nutritional information regarding all of my logged meals in the past week so that I can have a clear understanding of the health impacts of the food I am eating.

#	Description	Estimated Time	Owner
1	Create an UI panel to view a user's eating history, indicated through trends in macro consumption.	6 hrs	Aniket
2	Implement a corresponding API route for processing and sending meal-specific nutritional information.	4 hrs	Vaastav
3	Implement clearing of meal nutritional information weekly in the database.	3 hrs	Aniket
4	Manually test visual representation of nutritional information.	1 hr	Aniket

Completed:

The user nutritional summary UI screen, API route (supporting HTTPS encryption), and database schema were successfully implemented. Linking the UI screen to the API route took a little more time than expected, especially when handling errors and status codes returned by the HTTPS requests that were being made. The final product for this User Story well matches our mockup for the screen. Users can view their specific nutrition summaries for each week based on the aggregate totals from the meals that they have recorded.

User Story #3

As a user, I would like to be able to view the specific nutritional information for menu items.

#	Description	Estimated Time	Owner
1	Create a UI panel to view all of a menu item's nutritional information.	5 hrs	Aniket
2	Create a visual representation for caloric intake and macros of the menu item.	5 hrs	Aniket
3	Implement a corresponding API route for processing and sending meal-specific nutritional information.	4 hrs	Vaastav
4	Manually test meal nutrition functionality.	1 hrs	Aniket

Completed:

The menu item-specific UI screen, API route (supporting HTTPS encryption), and database schema were successfully implemented. Linking the UI screen to the API route took a little more time than expected, especially when handling errors and status codes returned by the HTTPS requests that were being made. The final product for this User Story well matches our mockup for the screen. Users can view their corresponding nutritional summaries for the week based on the meals that they recorded for that week.

User Story #4

As a user, I would like to be able to search for items on the menu page so that I can easily check for a specific item on the menu for the given dining court.

#	Description	Estimated Time	Owner
1	Create a search bar component for the user to look menu items up.	2 hrs	Anisha
2	Implement a function to render only the items that match what the user searches for	4 hrs	Anisha
3	Pre-process the menu items for the day into categories based on what allergen categories they fall under	3 hrs	Anisha
4	Render items correctly based on their allergen categories	2 hrs	Anisha
5	Manually test correct UI rendering and API Routes	1 hr	Anisha

Completed:

The search bar component was created to match the UI mockup and the functionality was successfully implemented through the use of a search bar package in react native. So, given that the user was searching up a menu item that existed at the time, it will render just that item or any items with the input text in its name. The menu items were also preprocessed into different data sets: all items, nut-free, dairy-free, gluten-free, and vegetarian. This was done by parsing through all of the menu items and checking various values in the body and adding them to the respective array. This was also done successfully as when the user selected one of those categories in the drop-down, only menu items with the given tags would be rendered on the screen.

User Story #5

As a user, I would like to upload a profile picture to my profile so that others can see me when finding a lunch buddy through the app.

#	Description	Estimated Time	Owner
1	Create a UI component to prompt the user for profile picture input.	1 hr	Anisha
2	Write a function to upload image files from the user's device to the application.	3 hrs	Anisha
3	Display uploaded picture as user's profile picture on profile page.	3 hrs	Anisha
4	Create API route to forward user profile picture to server.	2 hrs	Vaastav
5	Create a server function to save uploaded pictures to the database.	2 hrs	Anisha
6	Manually test uploading and storing images.	1 hrs	Anisha

Completed:

The UI component to prompt the user for their new profile picture was relatively easy to create and was completed quickly. In order to write the function to upload image files from the user's device, a package from expo was used called ImagePicker. Upon the user clicking the component to upload a photo this function would be called to access the user's device gallery where they could select the image they want as their profile picture. This would then instantaneously update the user's profile picture on their profile page for them to view. The API routes to get and set the user's profile picture were also implemented well as the new profile picture would persist upon logging out and back in of the user's account.

User Story #6

As a user, I would like the ability to make dining dollar transactions, as well as view these changes on my profile page.

#	Description	Estimated Time	Owner
1	Create UI to prompt user for dining dollar transaction input	3 hrs	Eric
2	Implement input validation on transaction amount	1 hr	Eric
3	Implement route to update dining dollar usage.	2 hrs	Eric
4	Implement UI so that dining dollar amount updates on profile page locally	2 hrs	Eric
5	Create unit test for UI and API route	2 hrs	Eric

Completed:

We successfully implemented the Dining Dollars transactions, a modal located on the profile page in which users could either add or subtract dollars from their account. The values were taken in by a text input and would be updated both in the database and locally. Also, invalid dollar amounts would not be accepted.

User Story #7

As a user, I would like the ability to make meal swipe transactions, as well as view these changes on my profile page.

#	Description	Estimated Time	Owner
1	Create UI to prompt user for meal swipe input	2 hrs	Eric
2	Implement input validation on transaction amount	1 hr	Eric
3	Implement swipe count so that it automatically resets at the end of each week	2 hrs	Eric
4	Implement route to update meal swipe usage.	2 hrs	Eric
5	Implement UI so that meal swipe amount updates on profile page locally	2 hrs	Eric
6	Create unit test for UI and API route	2 hrs	Eric

Completed:

We successfully implemented the Meal Swipe transactions, a modal located on the profile page in which users could subtract a meal from their account. The user would subtract a meal from their account by clicking a subtract button located on the modal. The value would be updated both locally and in the database, and if the user tried to change the number of swipes to be below zero, the program would not accept this. Also, the meal swipes automatically reset to their original amount at the end of every week.

User Story #8

As a user, I would like to have a meal history page so that I can view my past meal history.

#	Description	Estimated Time	Owner
1	Create a button on the profile page that allows users to navigate to the meal history page.	1 hr	Eric
2	Create UI page to display meal history information	3 hrs	Eric
3	Implement scrollable interface that allows user to view all past meal history	2 hrs	Eric
4	Implement routes to fetch meal information from database	3 hrs	Eric
5	Create unit test for UI and API route	1 hrs	Eric

Completed:

The meal history page was successfully implemented, allowing the user to view all of the past meals that they had eaten in the past as well as when they had eaten them. The meals are displayed in a scrollable list. Upon clicking any of the items, the user gets taken to the nutrition page for the item they clicked. If the user would like to see the meals displayed in order from most recent to least recent then they can click the sort button and the list order will be updated.

User Story #9

As a user, I would like to be able to mark meals as favorites so that I won't miss out on them when they are offered.

#	Description	Estimated Time	Owner
1	Create button on Profile page to redirect to user's favorite meals	1 hr	Sean
2	Create a UI panel for users to favorite menu items and view their current selection of favorite menu items	4 hrs	Sean
3	Have star icons show up next to menu items that are favorited.	3 hrs	Sean
5	Connect backend and frontend of Favorite Meal together for GET and POST requests	3 hrs	Sean
6	Create manual tests to evaluate the functionality of favorite meals gets, posts, and updates successfully.	1 hr	Sean

Completed:

We successfully implemented the Favorite Meals page where users can favorite their meals they enjoy at any Purdue dining facility. The screen is divided into two tabs, one for the user's current list of favorited meals, and the other tab with all of the favorite meals available. When the user selects one or multiple meals and favorites it, it initiates a POST request to our UserFavMeals API route.

User Story #10

As a user, I would like to customize my favorited meals to receive specific notifications.

#	Description	Estimated Time	Owner
1	Create notification UI with the user's current favorite meals list	5 hrs	Sean
2	Select specific items to turn notifications off or on	1 hr	Sean
3	Fetch the user's favorited meals and update the state of favorite meal route	4 hrs	Sean
4	Create manual tests for notification system	1 hr	Sean

Completed:

We successfully implemented the Notifications page where users can customize their notifications settings of their favorited meals. The screen is also divided into two tabs, one for the user's current list of favorited meals with notifications on, and the other tab with the user's current list of favorited meals with notifications off. On default, all of the user's favorited meal's notifications are set to on. After submitting their notification preferences, it initiates a POST request to our Notifications API route.

User Story #11

As a user, I would like to be able to remove favorite meals when my preference changes.

#	Description	Estimated Time	Owner
1	Create UI panel to have the current selection of user's favorite meals	3 hrs	Sean
2	Create remove button to remove selected meals in real-time	1 hr	Sean
3	Merge frontend and backend to allow for DELETE and GET requests	2 hrs	Sean
4	Create manual tests to test the functionality of the toggle feature.	1 hr	Sean

Completed:

We successfully implemented the Favorite Meals page where users can remove their favorite meals they now do not enjoy at a Purdue dining facility. The user can select one or more items they want to remove on their current list of favorited items and press remove. After the meals are successfully deleted in the UserFavMeals route, a Toast message pops up, confirming the deletion.

User Story #12

As a user, I would like to be able to input times that I am free to eat so that I am only recommended meals during those times.

#	Description	Estimated Time	Owner
1	Create a UI panel to enter a user's schedule of times that they are free to eat.	6 hrs	Aniket
2	Implement a corresponding API route for editing a user's schedule.	4 hrs	Vaastav
3	Configure database schema to store user account schedules.	2 hr	Aniket
4	Manually test scheduling functionality.	1 hrs	Aniket

Completed:

The schedule UI screen, API route (supporting HTTPS encryption), and database schema were successfully implemented. Linking the UI screen to the API route took a little more time than expected, especially when handling errors and status codes returned by the HTTPS requests that were being made. The final product for this User Story meets the acceptance criteria, allowing a user to input their schedule in one-hour increments for each day of the week.

User Story #13

As a user, I would like to be able to easily view the present day's meal options across Purdue's various dining facilities so that I may choose what menu items to eat from the options available.

#	Description	Estimated Time	Owner
1	Implement a cron job that loads the current day's menu to the database	2 hrs	Vaastav
2	Create a cron job schedule that causes the cron jobs to run every morning between 5 am - 7 am	2 hrs	Vaastav
3	Implement server function to ensure there are no duplicate menu items present or added to database	2 hr	Vaastav
4	Implement API route that allows the client to fetch the current day's menu for each dining court	4 hrs	Vaastav
6	Create unit tests for API route	2 hrs	Vaastav

Completed:

The data loading was implemented as a cloud function attached to three different cron jobs that ran every morning and performed the data loading. The jobs had to be divided into three due to timeout restrictions on individual jobs, forcing the total computation work to be spread across jobs. The data loader also successfully avoids duplicate menu item IDs ensuring uniqueness and a consistent database.

User Story #14

As a developer, I would like to construct a data preprocessing pipeline for our GNN model

#	Description	Estimated Time	Owner
1	Obtain and clean sample Kaggle data for preprocessing	2 hrs	Vaastav
2	Convert dataset into DGL graph	4 hrs	Vaastav
3	Assign feature set to created DGL graph	2 hrs	Vaastav
4	Implement test and train masks to pipe to GNN model	2 hrs	Vaastav

Completed:

The preprocessing pipeline involving three stages was successfully constructed and applied on our dataset of 50,000 recipes and 150,000 recipe reviews. The pipeline was designed in a manner to be completed independent of the file system and path it is being run on as it generates all the files required by the process on initialization and cleans up all the temporary files created front he file system

User Story #15

As a user, I would like to be given meal recommendations that are consistent with my past eating habits so that I can stick to eating foods that I enjoy and have eaten in the past.

#	Description	Estimated Time	Owner
1	Construct data pipeline from user database to GNN module	4 hrs	Vaastav
2	Create a predict function to serve recommendation requests	4 hrs	Vaastav
3	Implement API route to recommend meals	4 hrs	Vaastav
4	Create unit test for API route	2 hrs	Vaastav
5	**Total hours**	14 hrs	

<u>Completed</u>:

The GNN module was successfully constructed and operated as intended. Ihe module's computational pipeline involved creating a bipartite graph, training it to generate weights, exporting those weights and finally matrix multiplication using those weights to generate predictions. The predict functionality is accessible through our remote hosting link at the "/predict" route and will return a different response based on which user calls the route.

User Story #16

As a developer, I would like to create a dataset with dining menu logs and wait times so that I can use the data in our wait-time prediction modeling dataset.

#	Description	Estimated Time	Owner
1	Create function to merge historical wait-time dataset with historical menu item dataset	2 hrs	Mark
2	Create server function to fetch historical menu items from Purdue API	4 hrs	Mark
5	Conduct manual unit tests to verify proper implementation of data merge function	2 hrs	Mark

Completed:

Successfully retrieved datasets from Purdue API and Purdue dining app. Successfully inner joined dataset to have merged master dataset with no missing values and all appropriate variables. The end product is a master dataset that can be transferred straight into the prediction model for feature engineering.

User Story #17

As a developer, as part of the XGBoost modeling process, I would like to generate an importance matrix to identify the most important influencing factors affecting dining court wait-times so that I may use these factors for wait-time prediction.

#	Description	Estimated Time	Owner
1	Perform feature engineering to prep the dataset for modeling	4 hrs	Mark
2	Split dataset into train and test datasets and initialize hyperparameters and ML model	3 hr	Mark
3	Generate importance matrix, implement features into predictor algorithm	2 hrs	Mark

Completed:

From the master dataset generated in user story 16, we created time-series features like day, hour, week, month, etc. Columns with string values are converted to unique numerical values so that the model can read the values as categorical values. Data was split into train and test datasets by date, specifically August 1, 2020, as there was a huge gap in data from March 2020 to August 2020 due to COVID. As such, the gap was a convenient place for the split. The importance matrix showed that the time features were the most important, with the non-time factors (dining court location, menu item) being the least.

User Story #18

As a developer, I would like to fine-tune the XGBoost model to accurately predict wait-times per dining court.

#	Description	Estimated Time	Owner
1	Create XGBRegressor model with sample dataset and actual dataset	3 hrs	Mark
2	Calibrate hyperparameters to increase prediction accuracy	6 hrs	Mark
3	Visualize prediction results	2 hrs	Mark

Completed:

Running the dataset through the regressor model gave us a neat visualization of actual wait-time trends over time and how well our model predicted those wait-times, where we can clearly see traffic during lunch and dinner hours. Fine-tuning hyperparameters increased the accuracy of the model by about 30%. To transition to the next sprint where the model's results would be uploaded onto the app, a function was created that accepts a dining court and date and would return a plot of the estimated wait-time for the given dining court for the given date.

What did not go well?

Backend

In general, some of the original design plans for our API routes in the backend were not as robust as intended. The routing architecture caused poor performance on some of the frontend UI screens and slowed down navigation through the application. Additionally, we encountered multiple concurrency bugs that were introduced due to running multiple worker instances of the server program.

Frontend

In addition, the frontend experienced some issues as well, including multiple rendering issues due to data dependencies that took multiple hours to debug. Of the few user stories that these rendering issues impacted, the actual number of hours spent on these user stories exceeded the times that were originally allotted to the user stories in our Sprint 2 Planning document.

User Story #1

As a user, I would like to be able to see the correct menu items so that I can view what is offered at dining courts, record the right meal preferences, and record my meals.

1	Implement a function to fetch the right menu items with their allergen information based on the dining court.	2 hr	Anisha

Difficulties:

This task within user story 1 was difficult because of the way that the API routes were written. There was a lot of extraneous information being sent and it was difficult to access certain values that were passed to the frontend in the JSON object. This caused a lot of issues with dealing with null and undefined objects. So, we had to spend more time than was expected on this user story in order to reimplement the routes so that the fetches were easier to use. This then led to the last week of the sprint being extremely stressful and with a lot of work leftover.

User Story #9

As a user, I would like to be able to mark meals as favorites so that I won't miss out on them when they are offered.

4	Create an API route for the user's favorite meals (get, post, delete).	2 hrs	Vaastav

Difficulties:

We had to remodel our GET fetch for our UserFavMeals route. The problem that came up was that the user's favorite meal only contained the meal ID and not the meal name. This was a problem as to display the names of the favorite meals, we had to initiate an additional GET request to convert the meal ID into the name, and we had to do that for every single favorite meal that the user had. To elaborate, if a user had 100 favorite meals, the first fetch would get all 100 meal IDs. Then to convert the IDs into names, an additional 100 fetches would have been made. This ended up causing visibility issues. So, we had to rework the API routes to complete both ID and name fetches at the same time.

User Story #18

As a developer, I would like to fine-tune the XGBoost model to accurately predict wait-times per dining court.

4	Test and evaluate model accuracy	3 hrs	Mark

Difficulties:

The training and accuracy of the output of a model are highly dependent on the quality of the data that goes into the model. As such, the data we were given that includes the target feature, wait-time per dining court per date stamp, included only categorical values that were approximations of how long each line was. We had to attempt to generate numerical values of wait-times from these categorical segments by converting each categorical value to an appropriate numerical value. As such, the converted numerical values are still estimates and do not accurately represent the actual number of people in line. We used these estimated values to train and predict our model, thus our results would also not be as accurate as they could have been with more accurate data. Ultimately, however, the model was still able to successfully generate predictions that lunch and dinner spikes were about 80% accurate with hyperparameter tuning. However, ideally, we would like to have had more accurate target values.

How should we improve?

We did not achieve a perfectly even distribution of work per week, as we experienced lighter first and second weeks with a heavy third week. This was not due to procrastination but rather by the design of the sprint itself. For Sprint 3, we plan to incorporate more of the frontend in the backend meetings for creating API routes to ensure that our design is compatible with their intended implementation for the user stories at hand. While Sprint 2 was better than Sprint 1 in terms of workload distribution, we hope to spread the work more evenly during Sprint 3.

We could have done a better job of testing the parts of our application earlier. For the most part, we completed our parts on schedule, but we may have been able to save more time in the end if we did more documentation of testing each week instead of doing most of it in the third week. Additionally, it is easier to track the source of bugs when everything is well documented, so having information about bugs early on would be a good reference for the testing in later sprints. A good idea for the next sprint might be to start our testing document earlier and reference it continuously as we continue through the sprint.