Team 11

Purdue Eats

Team Members:

Aniket Agnihotri

Anisha Sinha

Eric Thompson

Mark Jin

Sean Joo

Vaastav Arora

PurdueEats

Sprint 1 Retrospective

What went well?

This sprint went fairly well for our team. We finished most of our user stories and were able to integrate each component such that a functioning, fluid PurdueEats prototype was created and presented by the conclusion of Sprint 1. Our user stories this sprint involved the following: registration and logging in functionality, profile pages, viewing dining facilities and their corresponding menus, ratings, and submitting meal preferences. All of these user stories function as specified in our Sprint 1 Planning Document and provide a good foundation to build out the rest of our application during Sprint 2 and Sprint 3.

User Story #1

As a user, I would like to be able to register for a PurdueEats account so that all of my information is associated with my account.

#	Description	Estimated Time	Owner
1	Create successive UI panels to enter account details.	3 hrs	Anisha
2	Implement a corresponding API route for user registration.	2 hrs	Mark
3	Configure database schema to store user account details.	2 hr	Vaastav
4	Create a function to encrypt user credentials before forwarding to API.	2 hrs	Anisha
5	Create tests for the UI panel and API registering functionality.	3 hr	Anisha

Completed:

The signup 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. Implementation of the successive nature of screens as mentioned in Task #1 was also difficult to get right and took longer than expected. The final product for this User Story well matches our mockup for the screen. User accounts can easily and securely be created with a corresponding username, email, password, and meal plan.

User Story #2

As a user, I would like to be able to login to my PurdueEats account so that I can track my meals and view wait times.

#	Description	Estimated Time	Owner
1	Create a UI panel to enter account username and password to begin login procedure.	2 hr	Anisha
2	Create UI panel routing from login panel to home page panel upon login validation.	1 hrs	Anisha
3	Create an algorithm to encrypt user credentials using secure hash algorithms (SHA).	3 hrs	Anisha
4	Create tests for UI panel and API login functionality.	3 hrs	Anisha

Completed:

The login UI screen, API route (supporting HTTPS encryption), and database schema were successfully implemented. Similar to User Story #1, 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. User accounts can easily and securely be implemented.

User Story #3

As a user, I would like to be able to enter my meal preferences so that I can have better meal recommendations based on my current likings.

#	Description	Estimated Time	Owner
1	Create a successive UI panel for meal preferences.	3 hrs	Aniket
2	Update API route corresponding to user's meal preferences.	2 hrs	Mark
3	Create a server function to save user preferences to the database.	2 hrs	Mark
4	Create unit tests to evaluate the UI panel, API route, server function and database transaction.	3 hrs	Aniket

Completed:

The UI panel for the meal preferences looks as it does in the mockup and the API route is implemented correctly so that the user preferences are saved in the database. The user can now easily submit meals that they would like to eat again. Selecting multiple items in the menu proved to be a little difficult but was solved using various packages. Overall, this user story and all of its tasks were successfully completed.

User Story #4

As a user, I would like to be able to easily view meal options across Purdue's various dining facilities so that I may choose where to eat from the options I'm given.

#	Description	Estimated Time	Owner
1	Create a UI panel to view all five dining facilities.	3 hrs	Aniket
2	Create a UI panel to view the menu items at a selected dining facility.	3 hrs	Aniket
3	Link pictures of each dining facility to each UI facility icon	2 hrs	Aniket
4	Implement API route to fetch menus of dining facilities for a particular day	2 hrs	Mark
5	Create unit tests for both UI panels and relevant API routes.	2 hrs	Aniket

Completed:

The UI panel for this user story was successfully created. It includes all 5 dining facilities: Earhart, Windsor, Wiley, Hillenbrand, and Ford as well as their images, names, and hours. It was difficult to format the images so that they would symmetrically fit on the screen but with the proper styling, it was able to be done. The route to get the menus of the dining facilities was also implemented well. In addition to that, since this UI panel actually is the host of two screens a tab selector at the top was created for a temporary homepage that will be implemented in a future sprint. The functionality for the right content to render based on the tab selected is also implemented well. The UI panel to view the menu items at a given dining facility was also created and looks like the mockup. Overall, this user story met all of its requirements.

User Story #5

As a user, I would like to be able to filter meals based on if they are vegetarian or contains specific allergens so that I can be sure that what I am eating fits my dietary needs.

#	Description	Estimated Time	Owner
1	Create a drop-down menu to select filterable options on the sequential menu's page	3 hrs	Anisha
2	Implement API route to fetch allergen information of specific menu items	2 hrs	Mark
3	Add allergen legend referencing allergen icons to the drop down menu	1 hr	Anisha
4	Create unit tests for drop-down menu, filtered results, and API implementation.	2 hrs	Vaastav

Completed:

This user story was initially easy to implement as the drop-down menu was created according to the mockups. However, it was difficult to actually render different items based on the selection in the drop-down menu. Using states and javascript within the rendering code we were able to satisfy this requirement. The allergen legend was created well through the use of modals. Overall, all the tasks in this user story were completed to satisfaction.

User Story #6

As a user, I would like to be able to rate meals so that I get better meal recommendations based on my preferences.

#	Description	Estimated Time	Owner
1	Create a “Record Meal” button on the sequential menus page that redirects to a record meal page.	1 hr	Sean
2	Create a record meal page that takes meal ratings as user input.	1 hr	Sean
3	Create an algorithm to package meal ratings input as a JSON object and send it to the server.	1 hr	Sean
4	Implement API route to receive meal ratings object and save to database	2 hrs	Sean
5	Create server function to forward meal ratings data	2 hrs	Sean
6	Create unit tests for “Record Meal” button, API routes for meal ratings and relevant database transactions.	2 hrs	Sean

Completed:

We successfully implemented a Record Meal page where users can record and rate their meals. Users can select menu items they have eaten, choose a rating of their choice, and submit their entry, which initiates a POST request to our MenuItemReviews API route. The entry is formatted as a JSON object and is stored in our database

User Story #7

As a user, I would like to be able to view a map of my current location and each dining court so that I can see directions to any dining court of choice.

#	Description	Estimated Time	Owner
1	Create a UI panel that shows a map with the user's current location	2 hrs	Sean
2	Add locations of each of the dining courts to the map component	2 hrs	Sean
3	Implement real-time updates of user location on map component	4 hrs	Sean
4	Create units tests for the UI panel and location updates	2 hrs	Sean

Completed:

Users can successfully view their current location and each dining court on the Map page. Each dining court is labeled with a marker, which contains the dining facility's name so that users can easily identify the five dining courts. Moreover, the Map UI uses Google Maps, which shows real-time traffic and where the user's device is pointing at.

User Story #8

As a user, I would like to be able to view the distance from my location to each of Purdue's dining facilities so that I may account for the distance to each facility as a factor in my dining decision-making.

#	Description	Estimated Time	Owner
1	Add distance to each dining facility marker on the map screen.	3 hrs	Aniket
2	Implement function to fetch user's current device location	2 hrs	Aniket
3	Implement a function to calculate distance from the user to each dining facility.	2 hrs	Aniket
4	Create tests for the UI panel, location fetching, and distance calculation	3 hrs	Aniket

<u>**Completed**</u>:

We successfully completed the fetching of the user's current location and each dining facility location so that users can see how far they are from each dining court. This information can be accessed when the user clicks on the five markers. Through this, users can now make better judgments on where they want to eat if the distance is a concern.

User Story #9

As a user, I would like to have a profile page so that I can edit my credentials, name, and meal plan as well as access my past meal history.

#	Description	Estimated Time	Owner
1	Create UI panel to display profile page information	2 hrs	Eric
2	Create an edit button so the user can change their name, meal plan, etc and have it updated in both the database and visually on the screen.	3 hrs	Eric
4	Link dining dollar transaction button and past meal history button to their designated pages.	1 hrs	Eric
5	Create an API route to obtain user's account data, name, and meal plan	2 hrs	Sean
6	Create unit tests for the UI panel and corresponding API functionality.	2 hrs	Eric

Completed:

The profile page is set up with all information visible. Edit buttons are available for changing information, and there is text that can be clicked to navigate to other pages. For right now a temporary profile picture is displayed since setting up profile pictures is a later story.

User Story #10

As a user, I would like to be able to change my account name or password so that I have control over my credentials.

#	Description	Estimated Time	Owner
1	Create a UI panel to enter a new account name or password.	1 hr	Aniket
2	Implement corresponding API routes for user account name or password amends.	2 hrs	Mark
3	Create an algorithm to encrypt user credentials using secure hash algorithms (SHA).	2 hrs	Anisha
4	Create server function to update user credentials based on new user data.	2 hrs	Mark
5	Create tests for the UI panel and corresponding API functionality.	3 hrs	Aniket

Completed:

The edit buttons for username and password changing bring up a modal that allows you to type in your new information, and upon clicking the back arrow sends that information to the database. The password is not visible locally so that can't be seen, but the username field is updated locally.

User Story #11

As a user, I would like to be able to change my meal plan and be able to view my dining dollars/swipes so that I have control over my credentials.

#	Description	Estimated Time	Owner
1	Create a drop down menu that allows for selection of dining plan	2 hrs	Eric
2	Create editable fields that store and display dining plan, dining dollars, and meal swipes	1 hr	Eric
3	Implement API route to send and receive meal information	2 hrs	Sean
4	Implement frontend fetch and post requests that update meal plan in the database and locally upon changes	2 hrs	Eric
5	Testing UI and API routes	2 hrs	Mark

Completed:

The edit button for the meal plan brings up a modal containing a drop-down menu, which allows for a choice between 4 different meal plans. Once a meal plan is selected your plan is updated locally and sent to the database, and the updated amount of dining dollars and swipes is retrieved from the database.

User Story #12
As a user, I would like to have a navigation bar so that I can access all of the features of the application from this screen.

#	Description	Estimated Time	Owner
1	Create a bottom tab navigation bar containing unique page icons.	2 hrs	Eric
2	Link navigation bar icons to designated pages.	3 hrs	Eric
3	Create unit tests for UI panel and application layout.	1 hr	Mark

Completed:
The navigation bar, though not a standalone screen, was successfully implemented with all intended logic and appearance. Linking the navigation bar icons to their respective pages was difficult to get right and took a little longer than expected given the navigation components used. The final product for this User Story well matches our mockup for the feature and functions as expected.

User Story #13

As a user, I would like my password to be reset when required so that I can still access my account if I forget my password.

#	**Description**	**Estimated Time**	**Owner**
1	Create a UI panel to reset password.	2 hrs	Eric
2	Create a link where users can pick a new password.	2 hrs	Eric
3	Implement an algorithm to send email from the app with reset password link.	2 hrs	Eric
4	Update API route corresponding to user credentials.	2 hrs	Sean
5	Create unit tests to evaluate if an email is sent and if the password updated successfully.	2 hrs	Eric

Completed:

We successfully created a button on the login screen that takes the user to a page that allows them to input their email. We successfully created a function that generated a random string to send to the designated email address. Upon the user entering the same string as the automated general string, the user was led to a page to reset the password. This password would be encrypted and updated in our database.

User Story #14

As a user, I would like to be able to delete my user account so that my user-generated data is removed from the application's database.

#	Description	Estimated Time	Owner
1	Add a button to the UI panel that would prompt the user to delete the account.	1 hr	Eric
2	Add a popup that would ask the user to confirm if they would like to delete the account.	1 hr	Eric
3	Implement deletion of account from database when user requests deletion.	3 hrs	Mark
4	Implement deletion of accounts from data structures when a user requests deletion.	3 hrs	Sean
5	Implement UI to change back to registration/sign-in page when an account is deleted.	2 hrs	Eric

Completed:

We successfully created a button on the user profile page that allows users to delete their accounts. The program would take the unique user id and subsequently delete records of the user from all tables in the database. The end state is that the user credentials are completely cleared from the database, even allowing a new user with the same credentials to be re-registered.

User Story #15

As a developer, I would like merge all API routes and remote deploy the application backend on Google Cloud Platform.

#	Description	Estimated Time	Owner
1	Change all route objects to APIRouter	2 hrs	Sean
2	Create central router and all auxiliary routes with respective prefixes	4 hrs	Anisha
3	Setup required backend environment on GCP app engine	3 hrs	Mark
4	Deploy and enable telemetry for application	3 hrs	Anisha

Completed:

On Google Cloud's appshot domain, our API is successfully accessible through the link **https://purdueeats-304919.uc.r.appspot.com/ .** Each subroute is accessible through this main route through its respective prefix. Each subroute has all of its REST routes document and testable through the /redoc and /docs prefix respectively.

User Story #16

As a developer, I would like to receive and record menu items from Purdue's dining menu website so that I can use the data as a factor in our wait-time prediction modeling dataset.

#	Description	Estimated Time	Owner
1	Implement API route to forward data from recorded user meals and ratings to the GNN module.	2 hrs	Vaastav
2	Create function to construct bipartite graph from user ratings fetched from server	4 hrs	Vaastav
3	Create function to add user nodes to graph when new users register	3 hrs	Vaastav
4	Create function to add graph edges based on user preferences entered after sign-up	3 hrs	Vaastav

Completed:

On new user registration, the user data is forwarded to the database by the API route for registration which also creates a node in the bipartite graph corresponding to that user. Similarly, on submitting a new menu item review, the review data is forwarded to the database by the API route for reviews which also creates an edge in the bipartite graph corresponding to that review.

User Story #17

As a developer, I would like to have custom GNN layer and model assets so that I can assemble a high performance GNN model engine

#	Description	Estimated Time	Owner
1	Implement network layers class of neural network	4 hrs	Vaastav
2	Implement node aggregation function.	4 hrs	Vaastav
3	Implement a stochastic gradient descent algorithm to train the model	3 hrs	Vaastav

Completed:

We implemented a working base layer class, a graph convolution layer class, and a relational convolution layer class. All these were chained together to produce our final model comprising 8 layers. The penultimate layer includes our node aggregation function that performs a heterogeneous dot product to produce inputs of the last layer. We also implemented a functioning training loop that ran up to 1000 epochs and successfully minimized loss.

User Story #18

As a developer, I would like to be able to visualize the model training process so that I can improve model performance based on new insights.

#	Description	Estimated Time	Owner
1	Plot GNN accuracy curve, loss curve, and model uncertainty, and store in database.	3 hrs	Mark
2	Create a blog of GNN learning results, feature edits, and hyperparameter edits.	3 hrs	Mark

<u>Completed:</u>

We successfully created a reusable function using ggplot that plots the epoch and loss for the implemented GNN models. This function will take in the output generated by the GNN model as a list and parse it into a database, which ggplot will then convert into a graph of loss per epoch. We successfully stored these logs into a csv file.

What did not go well?

In general, we were able to manually test several features exhaustively but failed to incorporate our testing criteria exhaustively. This caused us to spend much more time retesting implemented features every time they went through a minor modification or bug fix. Additionally, our graph neural network model did not have clear splits between training, testing, and validation data causing the model metrics to poorly reflect the model's actual performance and accuracy.

User Story #17

As a developer, I would like to have custom GNN layer and model assets so that I can assemble a high performance GNN model engine

5	Create unit tests for aggregator and nonlinear functions.	3 hrs	Vaastav

<u>Not Completed:</u>

We were able to train our model but did not perform a training/testing split for the dataset in advance. This caused us to use the same data points for our testing which led to the reported testing metrics poorly reflecting the model's actual performance and accuracy.

User Story #5

As a user, I would like to be able to filter meals based on if they are vegetarian or contain specific allergens so that I can be sure that what I am eating fits my dietary needs.

2	Implement API route to fetch allergen information of specific menu items	2 hrs	Mark

<u>Not Completed:</u>

Since the scheduled jobs that run daily to load menu items into our database were planned to be implemented in sprint 2, we had no existing menu items data to work with, and hence could not implement the route required to fetch allergen information due to missing infrastructure. We hope to fill in the gaps next sprint and implement both the daily scheduled jobs as well as the routes that depend on it.

How should we improve?

We did not achieve an 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 by the design of the sprint itself. We had a poor understanding and estimation of how many hours each user story was expected to take, and coincidentally the ones we underestimated the most were the ones for the third week. We would like to plan our future sprints for more even distributions of work, and we hope that sprint 1's experience can teach us how to better plan for future sprints.

Something else that we could do better is integrating our parts together more often. Throughout the sprint, we did a lot of our parts separately and were able to get them done, but once we had to put everything together we struggled. We lost a lot of time figuring out how to correctly combine the different components. If we had met up more often during the sprint to link up the parts of our project we likely would have saved a lot of time. A good idea for the next sprint might be to have weekly integration meetings so that everybody is caught up on all aspects of the project.