Team 15 - Sprint 3 Planning Document

Frederick Li, Shane Li, Shai Mohan, Joseph Singer, David Tong, Andrew Tully

Sprint Overview

Overview

For our final sprint, we will finish implementing our base functionality and look polish our project to look better and function more fluidly. Our backend team will be creating more functionality for cluster leaders and admins, and then move to assist the frontend team. The frontend team will be creating more page layouts and implementing functionality, as well as polishing our UI's. We anticipate most of our team's time to be devoted to frontend, because we are much newer to front end (more specifically JavaScript).

Risks/Challenges

The biggest issue for this sprint will be implementing functionality on the frontend. At this point, our backend is almost complete, but our frontend is still lacking some functionality. We will be moving some of our backend team members to frontend in order to combat the amount of work that needs to be finished. Some of us will be touching Javascript for the first time, but it is necessary that we are able to learn quickly from team members and outside sources. Because this is the last sprint, we need to delegate work efficiently and complete our work according to schedule.

Scrum Master: Shane Li
Meeting Schedule: Mondays, Wednesdays, and Fridays from 11:30 a.m. - 12:30 p.m.

Current Sprint Detail

User Story #1

As a logged-in user, I would like to be able to view information about a cluster and its members.

#	Task Description	Hours	Team	Owner
1	Implement routes to return user information	3	Back	Andrew
2	Implement buttons that will show user/cluster information in My Cluster page	7	Front	Joey
3	Implement API endpoints to return cluster information	3	Back	Andrew
4	Implement service layer methods to properly provide user/cluster information to controller layer	3	Back	Shane

Acceptance Criteria:

- Given that our backend and frontend can communicate correctly, when a logged in user views the “My Cluster” or dashboard page, then the user will be able to view the information about the clusters and its members.
- Given that the backend is implemented properly, when the frontend makes a call about cluster/user information, the proper information will be returned.
- Given that the database is implemented correctly, when the service layer makes a call to the database, the correct user/cluster information is returned for the service layer to prepare.

User Story #2

As a logged-in user, I would like to be able to sort cluster by newest or alphabetically by restaurant name.

#	Task Description	Hours	Team	Owner
1	Implement service layer methods to properly return a sorted list of clusters	2	Back	Shane
2	Implement API routes for specific sorting queries	3	Back	Andrew
3	Implement database calls to provide list of restaurants	4	Back	Fred
4	Implement database calls to provide a list of clusters	4	Back	Fred
5	Implement API routes to return an ordered list of clusters	3	Back	Andrew
6	Add button to select which type of sorting users want to use	8	Front	David
7	Generate view to display results and automatically refresh on sorting change	6	Front	Shai

Acceptance Criteria:

- Given that the backend and frontend are properly connected, then when a user selects a sorting method via a button on the view, then the corresponding list of sorted clusters will be displayed.
- Given that the database interface is set up correctly, when the service layer makes changes to sorting order, the order of clusters in the database will be updated.

User Story #3

As an administrator, I would like to be able to review order history.

#	Task Description	Hours	Team	Owner
1	Implement API routes for order history information	3	Back	Andrew
2	Implement service layer to properly save and return order history	3	Back	Shane
3	Implement database calls to update, save, and return orders regardless of completion	8	Back	Fred
4	Create search box to search for specific orders	4	Front	Joey
5	Create views to show order history	3	Front	Shai

Acceptance Criteria

- Given that our backend is implemented correctly, when a user requests order history, the frontend will make the corresponding API calls, and will return all order history to the user.
- Given that our frontend and backend communicate properly, and the backend functionality is implemented correctly, a user will be able to view their order history, as well as search through it via search bar.

User Story #4

As an administrator, I would like to be able to review cluster history.

#	Task Description	Hours	Team	Owner
1	Implement API to return cluster history	3	Back	Andrew
2	Implement database calls to save/return cluster history only if it is completed	9	Back	Fred
3	Create search box to properly search for a cluster	5	Front	David
4	Create views for displaying cluster history	2	Front	Shai

Acceptance Criteria

- Given that the backend implementation is completed, when an administrator makes an API call, then the cluster history will be returned.
- Given that the database is implemented correctly, when a Cluster is modified in the service layer, then its information will be stored correctly in the database.
- Given that the database is implemented correctly, when the service layer makes a call to request the cluster history, then the cluster history will be successfully retrieved and returned.
- Given that the frontend is implemented correctly, when an administrator visits the corresponding view, then the cluster history will be displayed.

User Story #5

As an administrator, I would like to be able to view total number of orders completed.

#	Task Description	Hours	Team	Owner
1	Implement API route for sending the total number of orders completed	3	Back	Andrew
2	Implement service layer methods to return number of orders completed	4	Back	Shane
4	Create views for displaying information related to the total number of orders completed	4	Front	Shai

Acceptance Criteria:

- Given that the backend is implemented correctly, when the frontend end calls the backend API, then the number of orders completed will be returned.
- Given that the database is implemented correctly, when the service layer makes a call the the database, the proper number of orders completed will be returned.
- Given that the frontend is implemented, when an administrator goes to the the admin page, then the number of orders completed will be shown.

User Story #6

As an administrator, I would like to be able to view total number of orders not completed.

#	Task Description	Hours	Team	Owner
1	Implement API route for sending the total number of orders not completed	3	Back	Andrew
2	Implement service layer methods to return number of orders not completed	4	Back	Shane
4	Create views for displaying information related to the total number of orders not completed	5	Front	David

Acceptance Criteria:

- Given that the backend is implemented correctly, when the frontend makes an API request, then the request is processed successfully and number of orders not completed is sent to the frontend.
- Given that the database interface is set up correctly, when the service layer makes changes to number of orders completed, the order information in the database will be updated.
- Given that the frontend is setup correctly, when a user visits the page about number of orders not completed, the order information will be successfully displayed.

User Story #7

As an administrator, I would like to be able to ban users.

#	Task Description	Hours	Team	Owner
1	Implement API call to ban users	3	Back	Andrew
2	Implement service layer to properly handle banned users and the ability to unban/ban users	4	Back	Shane
3	Create ban button for only cluster leaders to ban current members	4	Front	Shai
4	Create admin function to ban users	3	Front	Joey

Acceptance Criteria:

- Given that the backend is implemented correctly, when a user needs to be banned the backend properly handles the ban and sends that to the frontend to remove the user.
- Given that the database interface is set up correctly, when the service layer makes changes to banned users, the user information is removed.
- Given that the frontend is setup correctly, when a user is banned they are properly removed from clusters and groups.

User Story #8

As an administrator, I would like to be able to disband clusters.

#	Task Description	Hours	Team	Owner
1	Implement API requests to disband clusters	2	Back	Andrew
2	Implement logic in service layer to disband clusters	5	Back	Shane
3	Create admin page	6	Front	Joey
4	Add authentication feature in backend to detect whether user in admin	3	Back	Fred
5	Implement API request to send information to backend in frontend	2	Back	Andrew
6	Create buttons to select and disband clusters	4	Front	Joey

Acceptance Criteria:

- Given that the backend is implemented correctly, when the frontend makes an API request, then the request is processed successfully and the cluster is disbanded.
- Given that the database interface is setup correctly, when the service layer disbands a cluster the cluster should be removed from the database.
- Given that the frontend is setup correctly, when a cluster is removed, it should disappear off the list shown.

User Story #9

As an administrator, I would like to be able to view the number of clusters active.

#	Task Description	Hours	Team	Owner
1	Implement API call to return the amount of clusters active	2	Back	Andrew
2	Implement logic to serve API with necessary information	3	Back	Shane
3	Create view in admin page to display clusters active	8	Front	David
4	Implement API request to get information in frontend	4	Front	Shai

Acceptance Criteria:

- Given that the backend is implemented correctly, when the frontend makes an API request, then the request is processed successfully and the number of clusters active is increased or decreased.
- Given that the database interface is setup correctly, when the service layer makes changes to the number of active clusters, the number of clusters in the database will be updated.
- Given that the frontend is setup correctly, when a cluster number is changed, then the user will be see the updated amount of clusters.

User Story #10

As an administrator, I would like to be able to view the number of users online.

#	Task Description	Hours	Team	Owner
1	Implement API call to return the amount of users active	3	Back	Andrew
2	Implement logic to serve API with necessary information	3	Back	Shane
3	Create view in admin page to display amount of users active	5	Front	Joey
4	Implement API request to get information in frontend	3	Front	Shai

Acceptance Criteria:

- Given that the backend is implemented correctly, when the frontend makes an API request, then the request is processed successfully and the number of users online is increased or decreased.
- Given that the database interface is setup correctly, when the service layer makes changes to the number of online users, the number of users in the database will be updated.
- Given that the frontend is setup correctly, when a user number is changed, then the user will be see the updated amount of users.

User Story #11

As an administrator, I would like to be able to view user information.

#	Task Description	Hours	Team	Owner
1	Implement API call to return all user information	4	Back	Fred
2	Implement logic to serve API with necessary information	4	Back	Shane
3	Create view in admin page to display all user information	4	Front	David
4	Implement API request to get information in frontend	5	Front	Shai

Acceptance Criteria:

- Given that the backend is implemented correctly, when the frontend makes an API request, then the request is processed successfully and the user can view user information.
- Given that the database interface is setup correctly, when the service layer makes changes to the user information, the user information in the database will be updated.
- Given that the frontend is setup correctly, when a user's information is changed, then the user will be see the updated information.

Remaining Backlog

By the conclusion of Sprint 3, we intend to complete 26 of 26 user stories.

As a guest, I would like to be able to:

- ~~view existing clusters.~~
- ~~create an account using Facebook.~~

As a logged-in user, I would like to be able to:

- ~~view my profile.~~
- ~~view and join existing clusters.~~
- ~~sort clusters by newest, distance, or alphabetically by restaurant name.~~
- ~~view information about a cluster and its members.~~
- ~~request to start a cluster.~~
  ~~i) specify where I would like to pickup the order.~~
  ~~ii) choose a restaurant to order from.~~
  ~~iii) set minimum cluster size.~~
  ~~iv) set maximum cluster size.~~
  ~~v) set a duration time for the cluster.~~
- ~~log out.~~

As a cluster leader, I would like to be able to:

- ~~have all the capabilities of a cluster member.~~
- ~~approve/reject users joining cluster.~~
- ~~edit information about my cluster.~~

As a cluster member, I would like to be able to:

- ~~have all the capabilities of a logged-in user.~~
- ~~view restaurant information about my cluster.~~
- ~~view pickup/meeting location for my cluster.~~
- ~~view information about all other members of my cluster.~~
- ~~leave ratings for other members of my cluster.~~
- ~~leave a cluster.~~

As an administrator, I would like to be able to:

- ~~disband clusters.~~
- ~~review order history.~~
- ~~review cluster history.~~
- ~~view number of clusters active.~~
- ~~view number of users online.~~
- ~~view total number of orders completed.~~
- ~~view total number of orders not completed.~~
- ~~view user information.~~
- ~~ban users (if time permits).~~