cluster

Team 15 - Product Backlog

Frederick Li, Shane Li, Shai Mohan, Joseph Singer, David Tong, Andrew Tully
https://github.com/CS307-CLUSTER/Cluster

Problem Statement

When ordering food online, costly delivery fees and steep minimum order amounts often discourage potential customers from completing their order. If these delivery costs were reduced, customers would be more likely to place an order and businesses would benefit from the additional orders. Our project will offer a solution to this issue by connecting nearby customers to form large group purchases, reducing the delivery costs for each person that joins the group, or “cluster”.

Background Information

Audience:

Our target users are primarily students in Purdue residential halls, but also includes people living in apartments or staying in hotels. Our services have potential to eventually reach other campuses as well.

Similar Platforms:

- Seamless: Seamless is an application that allows coworkers to order together. It is specifically targeted towards companies that need to fulfill large group orders. Our application is marketed towards college communities, specifically residence halls and apartments on campus and is more focused on connecting users.
- Down To Lunch: Down to Lunch is similar to our application in that it groups people together. Its primary focus is helping people socialize and share a meal, while our platform’s motivation is to lower delivery costs and help meet price minimums, allowing users to decide whether they want to eat together.

Limitations:

- While using Seamless, you must already have a group of people that you are ready to order with. Our solution to this is allowing people to create or join a cluster. This system allows people to find others to order food with, regardless of prior association.
- Seamless is designed exclusively for businesses. Our project will provide a similar service to accommodate college students.
- Down To Lunch operates under the assumption that the group will meet at the restaurant. Our application will be used primarily for delivery groups.

Functional Requirements

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

- access the app via a website.
- login via Facebook.
- form a new cluster.
- set a minimum cluster size.
- set a maximum cluster size.
- specify which restaurant I want to order from.
- specify where I would like to pickup the order.
- remove a user from my cluster.
- find existing clusters.
- join an existing cluster.
- be redirected to hungryboiler website for ordering.
- view information about the restaurant.
- determine how far away the pickup point from me is.
- receive contact information of others in my cluster.
- rate other users.
- view my rating.
- mark my order as complete or incomplete.
- filter users allowed to join my cluster (if time permits).

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

- review order history.
- review cluster history.
- view number of clusters active.
- view number of users online.
- view number of orders completed.
- view number of orders incompleted.
- view user information.
- ban users (if time permits).

Non-Functional Requirements

Architecture

Frontend and backend will be separated for this application to allow easy integration, testing, and work allocation. This way, the frontend and backend will not have to rely on one another to do testing.

The backend will be based on a RESTful API (Representational State Transfer). It will be written in Java based on the Spring framework. The Spring framework is an enterprise-based infrastructure that provides support for Java-based applications. It will help us seamlessly integrate our data, service, and controller layers for the applications.

The frontend will be developed using Angular and BootStrap, which are HTML, CSS, and JavaScript frameworks for web development. The frontend will receive data by calling the RESTful APIs supported by the backend.

Scalability

Because we are using enterprise frameworks for this project, we will be able to scale our application easily. By separating our frontend and backend, we will be able to port our application to Android/iOS and other platforms more easily through our RESTful APIs.

Usability

The website should be convenient to navigate and easy to read. We must ensure that users will not be confused by our UI design. Color selection for the website is very important as we want our users to feel aesthetically satisfied. We must also ensure our website is accessible and usable from both desktop and mobile.

Response Time

We must be able to ensure low response times for both frontend and backend so users can have a responsive experience using our application.

Security

Security is not a high-risk factor for our application since we do not handle logins or payments. Most of the information we store will be for analytics and will not be sensitive information. We will ensure our client/server traffic and our backend is secure, and not prone to SQL injection etc.