FooCycle Project Specifications

Table of Contents

- Overview
- Breakdown
- Issues
- Timeline
- Features
    - MongoDB features
    - General features

FooCycle is going to be split up between **back-end** and **front-end** components. The back-end is going to primarily consist of MongoDB (our NoSQL database) and Express js, while the front-end will deal with connecting our back-end to the front-end. This will consist of Node js and React to build an interactive and user-friendly web application. Front-end frameworks are based in JavaScript.

Back-end - Vlad & Ryan

- **MongoDB** is a NoSQL database that we learned about in class. Great for **scalability** (especially for the large amount of data that we are going to have) and omits using schemas (unlike SQL; see **Potential Issues** below**).**
- **Express js** is a back-end framework that allows us to easily write code and build web applications and APIs.

Front-end - Hiren & Vivian

- **React** is a front-end framework based in JavaScript that allows users to build their user interface and design their web applications.
- **Node js** is a front-end framework that allows users to run their code on a server.

More information about the MERN stack can be found here:
https://www.geeksforgeeks.org/mern-stack/

Potential issues:

- When writing code in SQL and Python, one potential issue I encountered was data integrity - I was running into issues when writing into tables that reference other tables. This most likely won't be a problem in Mongo, but just something to keep an eye out for.
- I also encountered an error when trying to (for example) **add a user to a community that hadn't existed yet**. This should be an error we **can** handle.

Week 3/30 -

- Create small test code of inserting into MongoDB collections / database using Express js
- Build initial code that allows us to use MongoDB for our more specific functions like adding a user to a community, adding communities, etc.
- Start writing CRUD functionality outlined below
- Write bulk of back-end

Week 4/6 -

- Finish up writing CRUD functionality
- Finish up back-end
- Start writing front-end using Node and React
- Polish up back-end code
    - Testing back-end and front-end code
    - Testing client-side
- Finishing up building web application (front-end)

****features using MongoDB can be identified as either user-side or admin-side. User-side are features that any user can perform*****

Using MongoDB:

- **C**(reate)
    - Adding a community to our database (must go through admin-side if performed on user-side)
    - Adding a user to a community (user creates a profile)
    - Posting a new food item
- **R**(ead)
    - Viewing different communities (by any of its attributes i.e., zip code, name, etc.)
    - Viewing different postings within a community (by any of its attributes i.e., name, category, etc.)
    - Viewing different users (by any of its attributes i.e., user ID, name, etc.)
- **U**(pdate)
    - Update whether an item has been donated or not
    - Update basic profile attributes (i.e., age of user, name, etc.)
    - Update community names
- **D**(elete)
    - Delete user profile
    - Delete a community (not a priority)
    - Delete a posting

Other specific features to add:

- The (potential) distinction between users
    - **Foodbanks** (users who look for food)
    - **Donors** (users who are posting food items)
- **Authorization** / **Authentication** (is a Donor an authorized user by FooCycle to redistribute their food and are they trusted by other users and the company?)
- **In-app communication** - so users can coordinate within the app on how to arrange pickups, etc. rather than using external forms of contact