Christopher Edmunds

Project Proposal

1. Project Title: Lobster Bisque

2. Project Description:

There are not enough lobster bisque themed websites in the world. I am going to put an end to this injustice and make one myself!

My website is going to be a postboard in which a user can login to the website and post pictures of lobsters, bisque, or lobster bisque. The user will have the option to like, dislike, or comment on posts.

To ensure users adhere to the theme of the website, there would ideally be an AI that detects what the photo is and allows/denies the post. But alas, I do not specialize in AI, so I will have to have the posts submitted for review that a human can look over and verify if it follows the website theme requirements

This website will cater to the needs of those who are tired of being ridiculed for sharing their lobster, bisque, or lobster bisque on social media. This website will be a safe space to share their passions.

3. Features and Functionality:

Features:

1. Post Pictures
    a. Lobster Bisque is designed for users to share their passions for bisque. lobster, or lobster bisque. This includes posting pictures and leaving captions.

2. Reactions
    a. Lobster Bisque also allows the users to share how they feel about others posts too. This involves leaving comments, liking, and disliking posts.
3. Filter
    a. Lobster Bisque users will also have the ability to view and filter posts by number of likes, dislikes, and comments!
4. Profile
    a. Lobster Bisque users will have access to a profile page where they can view all of their posts, including likes, dislikes, and comments!
5. Anonymity
    a. Users will have the option not to login or create an account, and simply just peruse the website. Features like commenting, posting, liking, and disliking will be disabled for these anonymous users.

4. Database Information:

I plan to store user information, user account information, posts, and comments. The table will be set up as follows:

1. User:
    a. Account ID: Int
    b. UserName: String
    c. Password: String
2. Account:
    a. Account ID: Int
    b. Bio: String
    c. Profile Picture: Byte Array
    d. Post IDs: List[Post ID]
3. Posts:
    a. Post ID: Int
    b. Account ID: Int
    c. Comment IDs: List[Comment ID]
    d. Post Image: Byte Array
    e. Post Caption: String
    f. Likes: Int

    g. UserLikes: List[Account ID]
    h. Dislikes: Int
    i. UserDislikes: List[Account ID]
4. Comments:
    a. Comment ID: Int
    b. Account ID: Int
    c. Comment: String

UserLikes and UserDislikes is a way for the post to keep track of which user has liked or disliked it. This opens up the possibility to prevent users from liking or disliking it more than once

5. Database Integration:

I plan to use MongoDB to create the database for this website.
Lobster Bisque will use react hooks. These react hooks will take in input from the user and make respective api calls to a server that will either query and return data, or receive and push data to the database.

Firstly, Lobster Bisque will start on a login page, where you can login, create an account, or peruse anonymously.

If the user decides to surf anonymously, only the posts will be queried from the database through an api call to the server.

If the user logs in, the server will check against the database and get the corresponding account id. Once the user is logged in, they will be able to post, like, dislike, comment, and edit their account. All of these functions will be through making an api call to the server that will update the database

If the user decides to create an account, the user will be prompted to create a username and password, then the database will be updated and generate a new ID for the account, which will be returned through the api to the web app. The user will later be able to choose a profile picture and update their bio.

6.Technology Stack:

I plan to use the MERN Technology stack.
This involves using MongoDB for the database, handling the server side api calls to the database through an Express.js server, using React for the client side of the website, and Node.js for the runtime environment.

I'm most familiar with Express.js for server side processing because of this class, so I'll be using that to handle the database calls.

I'm also using React because it is the most fun and what I'm used to from both this class and because of my previous internship experience.

Because JavaScript is so easy to use and I'm most familiar with that, I'll be using that programming language for everything from the server side to the client side.

7.Credentials Management:

When it comes to the server accessing the database, I plan to pull the database key from a file on the server that is excluded in the git.ignore. That way the key won't be easily accessible to all those that find my repository on github.

For the client accessing the server through api, however, I plan on just letting that part be open. Since the user will not have the option to delete posts or comments, I do not need to worry about loss of data as the server will not even have the function to delete from the database. When it comes to a user adding posts, each post will be flagged for review to ensure compliance with the app theme in the first place. And if someone were to talk to the server directly, they'd need an account ID to make a post anyways

Development Plan

1. Project Timeline:

Start Date: November 12, 2023

End Date: November 26, 2023

Milestone 1 (11-12-2023): Client side rough draft

In this stage I plan to create a simple react app with mock data to display posts on a wall. The mock data will consist of posts and comments data and the structure of the data will reflect what the Database will look like.

Milestone 2 (11-15-2023): Client side cleanup

From here I'll set up the login page and account page with more mock data. At this point all the functionality for user interaction should be working and only the api implementation is remaining.

Milestone 3 (11-19-2023): Database and server setup

At this point I'll set up a MongoDB on my personal Laptop and start migrating the mock data for the client side into the Database.
Then I'll create an Express.js server that will handle interacting with the database and establish routes with specific functions to handle sending and receiving data from the database

Milestone 4 (11-22-2023): Server and client integration

From here All I should have to do is implement API calls using fetch in the react web app to communicate with the server in order to get and send data.
Once that is working I'll add some finishing touches and begin working on how to get this all running on azure.

Milestone 5 (11-26-2023): Migrating to Azure

This whole milestone will be dedicated to hosting the website, database, and API server in azure. I'll be working on figuring out how to host it first, then set up a domain that'll point traffic towards the React App and the Server, and then rewriting some react code to use the new Server address instead of localhost.
From here, I'll do some final testing to make sure everything works swimmingly.

2. Design and User Interface:

Lobster Bisque will be implementing a lot of color schemes relating to lobsters and bisque.

On the main postboard page, you can find a set of posts that display an image username and profile picture, caption, likes, dislikes, and comments. All with respective buttons. The post will be centered on the page with a nice cream color of some sort to complement the bisque colored background.

Once you click the comments button, you'll be redirected to a new page which displays only that post and the comments underneath it. From here you'll be able to add your own comments. The comments page will have a back button on the top left to exit back to the postboard view.

The postboard view will have a hamburger button on the right side of the screen that will open a dropdown selection view that allows the user to navigate to their account, settings where they can change their account profile, or logout of the app.

The postboard view will also include a filter section at the top of the page centered above the posts.

In the account page, users will see their profile picture followed by their name, their bio beneath that, and then the list of their posts beneath all of that.
In the settings page, users will be prompted with a form to change their username, password, bio, or profile picture. They can leave stuff blank if they don't want to change those fields.

3. Interactivity and Navigation

For this assignment, I will be making use of the react navigation npm package.

This will allow me to better control the navigation the user will go through and make the website have a better look and feel.

As I described earlier, the user will start on the main page once logged in. From here the user will be able to scroll and view all of the posts from other users and themselves.

The user will be given the option to like or dislike posts, or make comments. If the user decides to make a comment, they will be directed to a new view where only the relative post is visible followed by all of the comments for that post. From here, they can submit a comment themselves. There will be a back button to navigate back to the post view from here.

The user will also be able to access the settings, account page, and logout button from a hamburger button on the upper right of the main postboard

page. This hamburger button will be a dropdown of all the options, and each option will redirect the user to a new respective page.

In the account page, the user will see their username, profile picture, and bio, followed by a list of their posts that they can interact with the same as the postboard view.

In the settings page, the user will be able to change their username, password, profile picture, and bio.

The logout page will log the user out and redirect them to the login page.

Risks and challenges

1. Potential Risks:

a. Profile Picture:
    i. I think it may be a challenge to allow users to have profile pictures, as I somehow have to implement a way to let the user center the image inside a circle, which may be more work than it's worth\
b. Website Delay:
    i. The way I intend to sync the user interaction with the website and the database is through react hooks. This may cause a delay for the user after an input simply because the website is waiting for a database call to happen before updating

2. Mitigation Strategies:

a. Profile Picture:
    i. In this case, I may just try to make the profile icons square instead of circular.
    ii. If the previous method does not work, I'll just remove profile pictures entirely
b. Website Delay:

i. I do not intend to mitigate this problem, as the api calls should be relatively fast, and if there is a delay I'm sure the user will be patient when exploring the wonderful realm of lobster bisque