CEN 4010 Principles of Software Engineering, Summer 2023

Milestone 4: Beta Launch and Reviews

Project: Tastebudz

Tastebudz is an innovative web application that revolutionizes the way users create unique and delightful food recipes. TasteBudz aims to empower users by providing them with a personalized experience, enabling them to discover and craft delicious meals based on the ingredients they already have in their fridges. With the ability to create accounts and save recipes, TasteBudz will become a go-to platform for culinary enthusiasts, home cooks, and anyone else seeking inspiration in the kitchen.

Team: Group 8

Students:

Logan Karstens - lkarstens2021@fau.edu (Front End Developer)

Divyesh Mangapuram - dmangapuram2015@fau.edu (Back End Developer)

Colton Rohan - crohan2020@fau.edu (Scrum Master, Back End Developer)

Brandon Pojoga - bpojoga2019@fau.edu (Product Owner)

Rohit Varghese - rvarghese2021@fau.edu (Front End Developer)

7.25.23

Beta Launch and Reviews

Youtube : https://youtu.be/WFacJjSCJxg

Live Site : https://main.d2v9ll0tokuwya.amplifyapp.com/

History Table

Revision Date	Revision Description
7.22.23	Began working on the document and minor completion of some content
7.23.23	Major completion of document content
7.24.23	Further completion of content and majority of coding
7.25.23	Completion of content and Code reviews.

Product summary : TasteBudz

Welcome to a better way to a better way to find recipes. Have you ever had random ingredients in your fridge but did not know what to make? With our new website this common issue will be no more. Easily type in ingredients to our convenient search bar, and scroll through the many unique recipes this page has to offer. Our seamless integration of the Spoonacular API means you will never have to use another site again. With its vast number of recipes there is a meal waiting for everyone.

This is not just a site you use in a pinch. Making an account allows the user to scroll through recipes and save their favorites. Keeping track of recipes has never been easier. Each recipe comes with a thorough list of the ingredients you will need, the directions, as well as the nutrition facts. Save a healthy meal for today and a cheat meal for tomorrow. There are no limits to your culinary imagination. No other website matches this amount of content with our beautiful and user-friendly interface.

Looking ahead, TasteBudz will continue to innovate and expand to meet the evolving needs of its users. Personalization will be a key focus, with individualized meal recommendations and detailed nutritional analysis becoming even more prominent. Additionally, TasteBudz sees opportunities to integrate with smart kitchen appliances, enabling users to receive recipe suggestions and nutritional information directly through these devices.

Major Committed Functions

Priority 1

- Authentication: Users will be able to create and log into accounts using Firebase's authentication service.
- Search for Recipes: Users will be able to search by recipe name. Users will be able to see lists of popular or random recipes with basic filter functionality.
- Favorites: Users can favorite and unfavorite recipes. Users can view all of their favorite recipes in a centralized place. Users can easily and intuitively tell when a recipe is favorited.

Priority 2

- Recipe Information: Recipes can be easily viewed with step-by-step instructions for creating the meal. Recipes will include additional information like intolerances, ingredients, and nutrients.
- Advanced Search: Users can use dedicated search pages targeted for specific ingredients or nutrients. Users can enter a list of ingredients or nutrient constraints and find recipes that match that criteria.

Priority 3

- Community Interaction: Users can post comments on recipes. Users can like, dislike, or reply to other users' comments. Users have a feed on their homepage that shows recent activity related to their comments.

Unique Features

- Seamless login: Email/password will not be necessary on TasteBudz. All account-related information is tied to their linked google account. This also makes signing up just as fast as logging in a single time. Users will not need any additional account setup.

Product URL – https://main.d2v9ll0tokuwya.amplifyapp.com/

Usability test plan

Test Objectives

This usability test will be covering the search bar function within our application. The test will be conducted on a desktop computer with internet access and a modern web browser. Participants will access the website using the product URL on the previous pages.
Usability test objectives:

- Assess the ease of finding the search bar on the homepage.
- Evaluate the effectiveness and efficiency of the search bar in finding recipes based on specific ingredients.
- Ensure the results are accurate and from the spoonacular API.
- Verify that all recipes are correctly saved to the favorite list.

Test Scenarios

- **Search and Favorite**

  Starting Point: Participants begin on the homepage of the website.

  Task**:** Find the search bar, use it to find a recipe with a specific ingredient (e.g. "chicken"), and click the “Add to favorites” button on one of the results.

  Expected Outcome: Participants should successfully find a relevant recipe, favorite it, and receive some visual confirmation that it has been added to their favorites.
- **Favorite a random recipe**
- Starting Point: Participants begin on the homepage of the website.

  Task**:** Click the “Recipes” button in the navigation bar or the home page. Click the “Add to favorites” button on one of the results.

  Expected Outcome: Participants should successfully find a random recipe, favorite it, and receive some visual confirmation that it has been added to their favorites.
- **View and Remove Favorites**

  Starting Point: Participants are on any page of the website.

  Task: Access the favorite list and view the previously favorited recipes, and click the “Remove from favorites” button on one of the results.

  Expected Outcome: Participants should easily access the favorite list and see all the recipes they have favorited during the test.

Test Procedure:

- Introduce the test and explain its purpose and objectives to the participants.
- Ask participants to perform the three scenarios using the web application and the conditions described in the test objectives.
- Observe and record participants' actions, comments, and any usability issues they encounter. Ask participants for overall feedback on the website's search bar and favorite function

By following this test plan, we can make sure that the usability test is effective in evaluating the search bar functionality in the application. With the results of this test we can identify any places inside the search bar that might need improving before pushing the final product to the customer.

Questionnaire

Rate the following statements by checking the answer that most represents your experience while using TasteBudz:

1. The search bar was easy was it to locate and access for use:
    - ☐ Strongly Disagree
    - ☐ Disagree
    - ☐ Neutral
    - ☐ Agree
    - ☐ Strongly Agree

2. The search function provided accurate and relevant results:
    - ☐ Strongly Disagree
    - ☐ Disagree
    - ☐ Neutral
    - ☐ Agree
    - ☐ Strongly Agree

3. The favorite function worked as intended:
    - ☐ Strongly Disagree
    - ☐ Disagree
    - ☐ Neutral
    - ☐ Agree
    - ☐ Strongly Agree

Additional Feedback:

Please share any additional feedback you may have on the three features being tested.

_____

_____

_____

_____

_____

Your feedback is greatly appreciated.

QA test plan

Test Objectives: View Recipes

The objective for our usability test is to test the search bar and random recipe functions and ensure they are working effectively and as intended. One of the first objectives is to check the efficiency of the search bar, making sure it's only finding relevant and accurate search results for the user's search. This means we will be testing the APIs ability to retrieve relevant information based on the user's input as well as its accuracy in displaying the results. The second objective is to check the usability of the search bar to make sure there is ease of use and the function is user friendly. To test this we will check the APIs functionality with its interface, navigation, and user experience. The last objective is to identify any technical issues that may pop up during the usability test, such as a slow loading speed or compatibility issues. The goal of these objectives is to make sure that the search bar used within our webpage meets the needs and expectations of its users returning only the data they are requesting.

Hardware and Software Setup

The hardware and software used for the test plan are the necessary components required for testing the recipes search bar functionality. The hardware used for the test will be a personal computer with an internet connection, keyboard, and mouse. The software required for testing includes most modern day web browsers such as Firefox and Chrome, we do this to test the accessibility of the search bar across different web browsers. Another piece of hardware used throughout the tests will be mobile devices to evaluate its responsiveness and compatibility with various devices. The primary devices used for testing will include a computer and a smartphone. The main goal of this setup is to make sure that the search bar is tested under different scenarios and to identify any compatibility issues that may arise on different devices and web browsers.

Feature to be Tested

The feature that is going to be tested in this test plan is the recipe search bar function. The search bar is a crucial feature that allows users to search for recipes based on recipe names or key terms. This function is expected to provide accurate and relevant recipe results. The test cases will include searching for recipes using various keywords and ingredients to evaluate the search engine's accuracy and relevance of search results. The test plan will also evaluate the random recipe feature that displays six random recipes to the user.

Actual Test Cases and Results

1. **Search for a recipe by its name**
    - **Input:** User inputs the name of a recipe in the search bar and searches based on the input
    - **Expected Output:** the search results page should show the recipe typed in and all of its other information to go along with the recipe
    - **Actual Output:** the search worked as intend and provided the user with the proper information based on the recipe they typed in
2. **Show random recipes**
    - **Input:** User selects the recipes button in the navigation bar or on the homepage.
    - **Expected Output:** A list of six random recipe names, images, and preparation times will be displayed on the screen after a brief loading period.
    - **Actual Output:** The feature worked as intended and the user saw the six recipes. Two of the preparation times were “Untitled” because they were not specified on the spoonacular API.
3. **Add one of the searched recipes to a favorites list**
    - **Input:** User with a recipe on their screen clicks the “Add to favorites” button on the recipe’s interface.
    - **Expected Output:** The user should receive some visual confirmation that the recipe was added to their favorites. They should be able to see their favorites on a separate page.
    - **Actual Output:** The user was presented with a notification saying that the recipe was added to their favorites. The user then found their list of favorites using the navigation bar.

Test Summary

The tests were carried out on two different web browsers, Chrome and Firefox. The main aim of the test was to make sure the search bar function performed accurately and efficiently on both browsers without any issues. All the test cases that were performed ran successfully, and the expected outputs were what was expected. Based on the testing it can be concluded that the search bar function is working effectively and meeting the needs of the users. this QA Test Plan helps improve the search bar's user-friendliness and functionality, leading to a better user experience in the end.

Code Review

When creating this project, we used next.js with React, and as a result, we followed the common conventions associated with the framework. The coding style we used throughout the project was component-based architecture. We also utilized tailwind CSS to help simplify the styling process. Provided below is the code we used for the frontend/backend of our search bar, showing its implementation as well as its functionality.

Frontend Searchbar component

```
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome'
import { faSearch } from '@fortawesome/free-solid-svg-icons'
const SearchBar = ({ value, setValue, onSearch, className}) => {
  const keyDownHandler = (e) => {
    if (e.code === "Enter") {
      onSearch();
    }
  }
    return (
      <div className={className}>
        <div className="relative">
          <div
            className="bg-transparent border-black focus-within:border-b-2 p-1"
          >
            <FontAwesomeIcon icon={faSearch} className="pe-2" />
            <input type="text" value={value} onChange={e => setValue(e.target.value)}
onKeyDown={keyDownHandler} placeholder="Find recipe..." className="bg-transparent
placeholder:text-black placeholder:italic focus:outline-none"></input>
          </div>
        </div>
      </div>
    );
  };

  export default SearchBar;
```

Backend Search Functionality (Cut down)

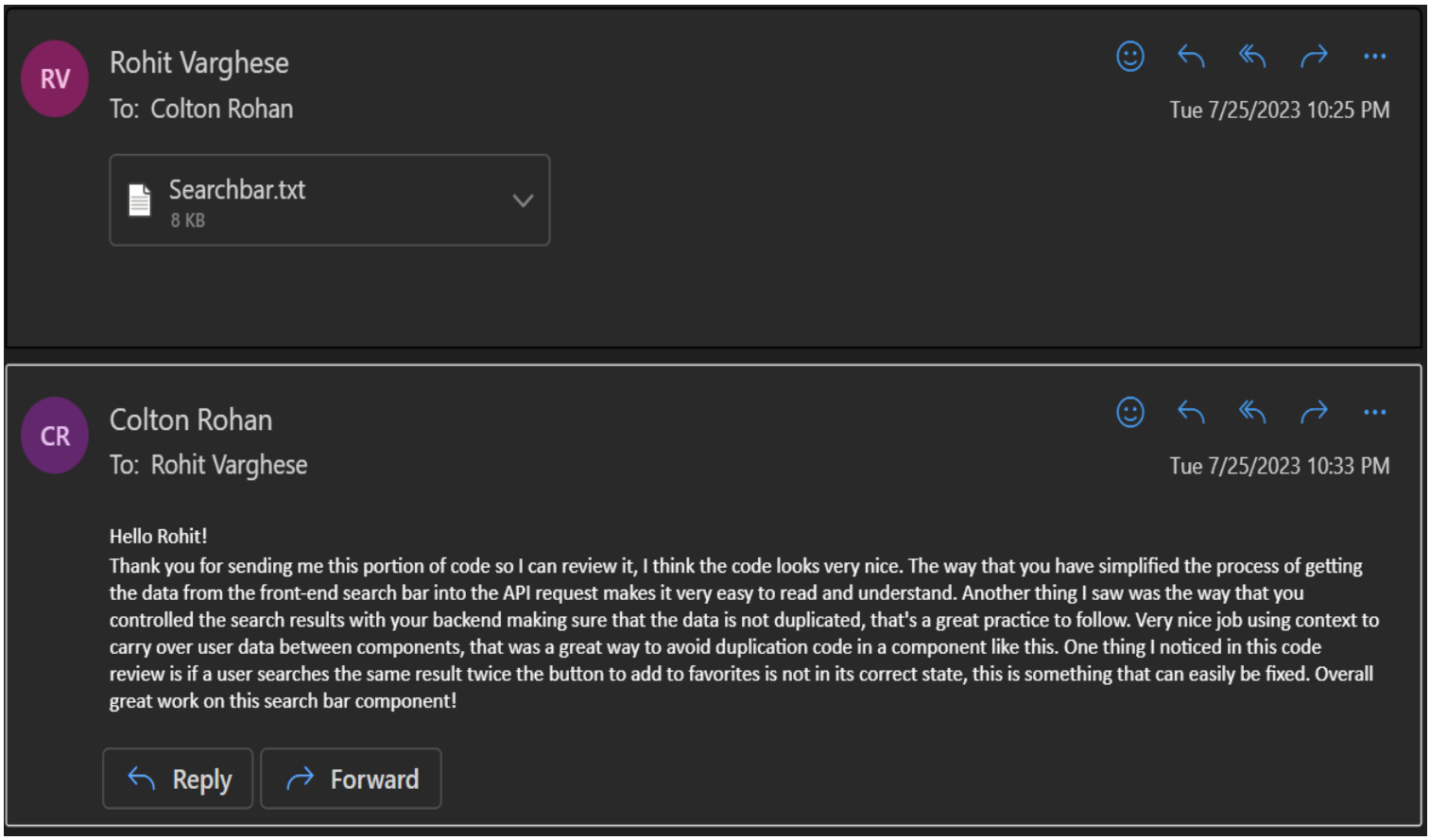
```
const Recipes = () => {
  const ctx = useContext(authContext);
  const [recipes, setRecipes] = useState(null);
  const searchParams = useSearchParams();
  const search = searchParams.get("s");
  const [searchTerm, setSearchTerm] = useState(search ? search : '');
  const router = useRouter();

  useEffect(() => {
    const searchTermHandler = () => {
      setSearchTerm(router.query.s);
    };
    router.events.on("routeChangeComplete", searchTermHandler);
    return () => {
      router.events.off("routeChangeComplete", searchTermHandler);
    };
  }, [router.query]);
  useEffect(() => {
    const getData = async () => {
      if (!searchTerm) {
        setRecipes([]);
        return;
      }

        // get recipes by search term
        const result = await axios.get(`../api/recipes/name?s=${search}`);
        result.data.results = result.data.results.map((recipe) => ({
          ...recipe,
          loading: false,
          favorite:
            userFavorites.find((f) => f.recipe_id == recipe.id) !== undefined,
        }));
        setRecipes(result.data.results);
      }
    };
    const timeout = setTimeout(getData, 100);

    return () => clearTimeout(timeout);
  }, [ctx.loading, searchTerm]);

  useEffect(() => {
    console.log(recipes);
  }, [recipes]);
```


RV
Rohit Varghese
To: Colton Rohan
Tue 7/25/2023 10:25 PM
Searchbar.txt
8 KB
CR
Colton Rohan
To: Rohit Varghese
Tue 7/25/2023 10:33 PM
Hello Rohit!
Thank you for sending me this portion of code so I can review it, I think the code looks very nice. The way that you have simplified the process of getting the data from the front-end search bar into the API request makes it very easy to read and understand. Another thing I saw was the way that you controlled the search results with your backend making sure that the data is not duplicated, that's a great practice to follow. Very nice job using context to carry over user data between components, that was a great way to avoid duplication code in a component like this. One thing I noticed in this code review is if a user searches the same result twice the button to add to favorites is not in its correct state, this is something that can easily be fixed. Overall great work on this search bar component!
Reply
Forward

Self-check on best practices for security

There are four main assets that our group will be protecting– the interactive frontend, the database, API's, and user login. Authentication is the most important part of the website and will dictate a user's experience with the website. A user who is not authenticated with the website will have a severely limited experience and access such as not being able to interact with the community or adding favorites. The latter is very important as we do not want any unauthorized or spurious data committed into the database to keep it manageable and appropriate. In Github, the database connection is stored in a config.js file but does not include the username and password for the database. These will be stored in a .env file that each developer has on their local machine.

The API's will also be protected in order to prevent bad actors from using applications such as Postman to get information using the endpoints. There are two checks in place to prevent this from occurring. An API key is generated for use of the developers and will also be stored locally on the .env file. Similarly, API's will also require users to be authorized and authenticated. Lastly, user data is very important to secure. Using firebase authentication services, passwords are not stored in plain text. Rather, firebase uses a method known as "hashing" which takes a password and transforms it into an string of characters using a one way has function. Incorporating these multi-layered security measures will make sure our users get the best experience with our website and instill confidence that everything is secure.

Self-check: Adherence to original Non-functional specs

Simple Design	**ON TRACK**
Manageable Performance	**ON TRACK**
Security	**COMPLETED**
Accessibility	**ON TRACK**
Scalability	**ON TRACK**
Reliability	**COMPLETED**

Simple Design: As the application is progressing we are sticking to a simplistic design, making it easy for users to navigate the web page. The navigation bar interface that we have created allows users to interact with the entire website through the different routes. We plan to add more to the design before the final release of the application. **ON TRACK**

Manageable Performance: The way that we are loading data for our components is proving to be very beneficial to load speeds. The application loads the data from the APIs with fast speeds and low latency. The connection to the database is set up flawlessly meaning adding data based on individual users is extremely efficient. We are continuously working on making sure the loadtimes within the application are low and have little effect on the user experience. **ON TRACK**

Security: Security was a top priority when creating this application, that being said we made sure to deal with any potential security issues. The application stores all of the sensitive information such as API keys in environment variables, preventing any malicious users from obtaining the site's keys. The user data is stored through firebase so any sensitive information that might come along with the google sign in option is already protected by the firebase services. **COMPLETED**

Reliability: TasteBudz should be available for use 24 hours a day, 7 days a week. There should be little to no downtime in the application, the live site will hold an older version of the webpage until the newest version is pushed to github. It will be up and running as long as the AWS amplify server is online. **COMPLETED**