System and Unit Test Report
Delish, Foodiez, December 8th, 2020

System Test Scenarios

Sprint 1:

- User story 1 from sprint 1: As a user, I would like to have a menu of different cuisine types in order to have a variety to choose from.
- User story 2 from sprint 1: As a user, I would like my search area on the website to begin where I am in order to save time searching for restaurants.
- User story 3 from sprint 1: As a user, I would like to have restaurants denoted by pins on the map so it is easy to begin viewing restaurant information.

  Scenario:

  1. Start the webpage;
     - Allow location
       1. Map should center at users location
       2. If not allowed, error box will populate
     - Click on the Menu button
     - Menu with cuisine types folds out from the left
     - View cuisine types
     - Click on close menu button
     - Menu closes/folds back to the left

Sprint 2:

- User story 1 from sprint 2: As a hungry person, I would like to be able to filter restaurants based on types of cuisine, from the menu, in order to find the food(s) I want.
- As a user, I would like to be prompted with an intro message to know what the features are and how to utilize the web app. (3) (Cindy)

  Scenario:

  1. Start the webpage
     - Read and close intro pop-up
     - Allow location
       1. Map should center at users location
       2. If not allowed, error box will populate
     - Click an cuisine type
     - The map should populate with restaurants associated to cuisine type
     - Click the same cuisine type
     - The map should clear and repopulate with restaurants associated to cuisine type
     - Click another cuisine type
     - The map should get rid of existing markers and populate with restaurants associated to the new cuisine type

Sprint 3:

- User Story 3 from Sprint 3: As a user I would like to see clear Clustering of restaurants icons so the map does not get
- User Story 4 from sprint 3: As a user I would like to select multiple types of cuisines in order to compare restaurants.too cluttered.

  Scenario:

  1. Start the webpage
     - Read and close intro pop-up
     - Allow location
       i. Map should center at users location
       ii. If not allowed, error box will populate
     - Click on Menu
     - Click on Cuisine type
     - The map populates with restaurants of denoted cuisine type
     - Click on additional Cuisine type
     - The map populates with the addition restaurants from the new cuisine type
     - Click on the first cuisine type again
     - The map should get rid of the restaurants that were showing for that cuisine type (i.e. checkbox logic)

Sprint 4:

- User story 1 from sprint 4: As a user, I would like to be able to view info boxes with restaurant info for each restaurant displayed on the map.
- User story 2 from sprint 4: As a user, I would like to be able to connect to Delish on the web to use it in my browser.
- User story 4 from Sprint 4: As a user I would like to see restaurant icons to distinguish restaurants by cuisine type on the map.

Scenario:

1. Start the webpage
   - Read and close intro pop-up
   - Allow location
     i. Map should center at users location
     ii. If not allowed, error box will populate
   - Click on Menu
   - Click on Cuisine Type
   - The map populates with restaurants of denoted cuisine type
   - Click on a restaurant icon
   - Info box with name, address, website/search link, and (if available) photo of restaurant pops up above the selected restaurant
     i. Click on website/search link to be taken to restaurants website or search

    - Click on close button info box
    - Closes info box associated with selected restaurant

Unit Tests

- Will
    - Tested the backend API by using Postman to send different post/get/put requests to and from the server/database to verify documents can be inserted, updated, and found correctly and that the endpoints work
    - Test upvote and downvote logic by creating buttons to interact with the respective document stored in the database by either incrementing the rating value by 1 or -1 with each click of the respective button
    - Tested the server.js file would properly host a server locally during development and allow access to the different endpoints created by using Postman and opening a browser
    - Tested unifying the frontend to the backend using AJAX calls and lots of console.logs
- Cindy
    - Tested Geolocation by allowing/blocking location and verifying the map
    - Tested pop-up by running HTML file and verifying if it showed/closed with the already implemented code
    - Tested clustering of icons by zooming in/out of the map
    - Tested menu box/Delish box by running HTML and CSS and verifying inspect object
    - Tested images showing up on info box by selecting different cuisines

- Cyrus
    - Tested the ratings by logging HTTP status codes and verifying that actions on the frontend affected the backend
    - Verified that local website rendered correctly on Firebase (GCP)
    - Test that the API spec (openAPI) matches API intended functionality
- Wei
    - Tested that info boxes were working properly and displayed the correct information.
    - Tested rating buttons were able to send requests to the backend.
    - Tested UI for correctness and making sure everything looks correct and sizing and fonts are visually appealing and acceptable.
- Andrew
    - When working on and testing the cuisine search menu logic modules in cuisineSearch.js and search querying results I did tests involving sequencing of selecting and removing certain cuisine types (via clicking on and off from the given types). From this I was able to come across various bugs in the form of markers which were on screen when their cuisine type did not appear selected as well as bugs such as markers loading in non-local areas due to the search going out of the local area when it could not find many restaurants of the selected type in the users local area.
    - In the placeDetails.js setMarkerInfoBox module Cyrus and I utilized console.logs and promises to ensure the module had to be run first before calling on create marker. This ensured the synchronicity of a place finding it's rating if stored in the database/adding itself to the database and then creating the marker after having that information.