CAPSTONE 4ZP6B PROJECT

Verification & Validation Plan

Version 0

FINDDIT

GROUP 19

Prakarsh Kamal

Japnit Singh

Kanwar Sandhu

Hriday Jham

Purpose of the Project

Purpose:

The purpose of building this application is to enhance the social experience and decision-making process for groups of users when it comes to choosing a restaurant. The app aims to increase end-user happiness when making a collective decision in a group by streamlining the decision-making process through an interactive stack-based swipeable UI and a cost optimization function considering user preferences. The goals of building this application are to increase user happiness, reduce decision-making fatigue, and facilitate enjoyable group outings through an interactive application.

Component Test Plan

1. Out-of-Scope Objectives

The following objectives and requirements are out of scope regarding verification and validation.

- Third-party library behavior testing
- External API's reliability
- Legal standards and compliance testing
- Device-specific testing

1.1 Unit Tests

Component	Testing strategy (front-end)	Testing strategy (back-end)
User authentication component	• Unit test with Jest to validate that login/signup buttons are triggering the correct functions • Unit test with Jest that expects create/join session	• Unit test with mocked database call to ensure signup function request is received with the right parameters and returns expected status code

	buttons to be visible after the user is authenticated	
Group creation component	● Unit test with Jest to validate that clicking “Create Session” loads group creation components ● Unit test with Jest to validate group is not created until all fields are filled	● Unit test with mocked database call to ensure group creation function request returns expected status code with the correct parameters.
Restaurant stack component	● Unit test with Jest to check a non-empty card array is rendered. ● Unit test with Jest to ensure the length of the rendered card array is correct is the same as the input length	● Unit test with mocked API call is called with correct parameters and request returns expected status code
Session history component	● Unit test with Jest to ensure the session array is rendered with clickable buttons to join active sessions.	● Unit test with mocked database call to ensure session history request returns expected status code when function called with correct parameters.
Leaderboard component	● Unit test to ensure the leaderboard array is rendered in the same order/restaurants as the input array	● Unit test with mocked database call to ensure leaderboard request returns expected status code
Scoring algorithm component	N/A	● Unit test with required preferences gives expected restaurant ranking

We will do unit tests for both the frontend and backend features.

Frontend

- We will be using Jest for unit tests and testing various components on the front end. This includes buttons, modals, input fields, switches, etc. We will ensure these interactable components exist. To achieve this, we will attach 'data-testid' to the components making them available for spec file testing.
- For testing API calls and components, we will use 'mock' in jest to simulate the expected happy behavior for the component which depends on other components/API calls.

Backend

- We will be mocking the database calls made for various components and assert if, with the correct parameters, they are sending back the expected status code and phrase.

1.2 Performance Tests

Component	**Testing strategy (Manual)**
User authentication component	● Measure response times for authentication components ensuring they are below 100 milliseconds
Group creation component	● Measure response time for group creation by adding an arbitrarily large number of users to a group
Restaurant stack component	● Measuring the swiping responsiveness and animation of the restaurant card stack is smooth
Session history component	● Measure response time for session history by creating an arbitrarily large number of sessions
Leaderboard component	● Measure response time for the leaderboard component ensuring it is below 100 milliseconds

Scoring algorithm component	N/A

Performance Tests & Metrics

We will do performance tests which test the app's load and response times. The aim is to keep the app fast and have a response time of around 100 milliseconds. This will be achieved by manual performance testing on multiple mobile devices and checking the latency.