Test Plan

By Benson Gao, Nick McDaniel,
Michael Ly, and Julian Tran

Table of Contents

What We Are Testing

There are quite a few features that need to be tested in depth to ensure that our system works properly and creates an ideal user experience. The list of items that we will be testing is as follows: Login Credentials, Chat/Messaging, Location/Guide Discovery Ranking Algorithms, Server Latency, API Integration, Profile Modifications, and the Report System. The reason that these are the features we chose to test is that these are the features that have been decided as the most necessary to create a working app that provides a good user experience.

There are a few high priority features we need to test as these build the basic function of out application. If they do not work properly, then the app will not function at all as intended. Login Credentials are important to test as this application is a form of social media. We want users to be able to use their profiles as well as keep any of their information that they entrust with us secure. Chat/Messaging is another high priority feature to test. The reason for this is that the main functionality of the app is to get users to talk to each other in order to gain otherwise unknown information about locations and events as well as build connections. The last high priority feature being tested are the discovery algorithms. Our system needs to recommend good locations, events, and guides to users. Otherwise, the app will not be seen as useful and user retention will drop off drastically.

The other features listed before are not high priority features as they are not essential to the basic function of the system. However, we found them still important enough to stress test as they will improve the overall experience of the application. The purpose of testing Server Latency is to make sure that users will have access to the information that they want when they want it. It is important to make sure that the servers that we are utilizing do not crash or slow down heavily when many users are on the platform. API integration is important to test as well as our locations shall be suggested using the Yelp API. Ratings that are displayed on the locations page will come from the Yelp page, therefore correctly integrating the API and testing its functionality within our app will lead to more accurate suggestions. Profile Modifications are another feature we are choosing to test thoroughly. The reason for this is it is what the system will be basing its recommendations off of. Therefore, making sure that the profile settings lead to accurate results is also important to a better user experience. Finally, the last item that has been chosen to be tested is the report system. Since we are making a social media app, making sure that users that act inappropriately are punished accordingly is a necessity as it keeps those who are using the platform with good intentions have a better time within the app.

What We Are Not Testing

The aspects of our system that we won't be testing will be the UI elements, rating a location/guide, and the server capacity under large loads. The UI elements will receive a basic amount of testing to see if UI functions as intended but in depth testing is not needed. Similarly, rating a location/guide will not be strictly tested because our UI limits user inputs to a few controlled options. We also will not be testing the server capacity under large loads due to the resources required as well as our inability to realistically change or improve the system. We are

relying on Firebase for our server so if there are issues, the only feasible change we can make would be to upgrade the tier of our account with them.

Login Credentials

When testing User Logins, we want to make sure that our system handles a variety of invalid characters and inputs properly. We'll be testing for valid/invalid username and password combinations, incompatible characters such as foreign characters, already occupied usernames, and SQL injection attacks. We'll also be testing our password hashing system. Testing this aspect of our application is important not just to reduce the number of bugs but also to ensure our system and users are secure from attack.

We will be testing by using a series of unit tests that handle the wide array of problematic inputs for username and passwords. This will allow us to prevent as many issues as possible as well as being easily expanded for new cases/situations. The Pass criteria for these tests will be that valid username/password combinations lead to successful connection to the service and invalid username/password combinations lead to a proper error message. Fail conditions are when invalid username/password combinations lead to a successful connection to the service, invalid characters are accepted, username or password exceeds limit and causes overflow, and SQL injections are allowed to execute.

Chat/Messaging

The goal of our application is to create an environment that promotes user connections and user interactions with each other. This portion of the project needs to work with minimal bugs because of this. Therefore, we will carry out detailed stress tests relating to users sending messages to other users, receiving messages from other users, and making sure that users are properly notified when they have been contacted. With a functional chat system, we can increase user engagement and activity which in turn can lead to the growth of our user base.

To perform our testing on this portion of the project, we will write scripts that will attempt to send messages of varying lengths and at a varying frequency to make sure that all of them are received by the intended user. We will also be checking to see that the user receives proper notifications in a number of scenarios. Examples of these scenarios would include being notified of a message while not actively looking at the chat the message is being received from, being notified while the app is closed, being notified while the user is using another app on their device, and being notified while the users device is locked.

The passing criteria for this test is if the user receives all intended messages within an acceptable time frame and they are notified of the messages that they receive, no matter how they may be using their device at the moment of message receival. Failure would be any scenario that does not lead to this. This includes not being notified of all messages, not receiving all of the messages sent to a user, or receiving messages manner that is slowed or otherwise altered.

Location/Guide Discovery Ranking Algorithms

The location and guide discovery ranking algorithms will be a key portion of our application and very important to the user's perception of the usefulness of our application. Therefore, we want to make sure that our tests allow us to confirm that the way locations and guides are listed are as expected. We'll be testing a wide variety of example locations with different parameters and how they match different sets of user preferences for location discovery. Similarly for guide discovery, we'll be testing guides with a different percentage of matched interests. We also want to test for locations/guides with zero reviews and ensure that they receive a proper amount of exposure. For guides specifically, it will be important to also account for guides that haven't logged in for a significant period of time. We would not want well rated guides who have quit using our applications to continuously appear at the top of recommended lists.

The Pass conditions for the tests we'll perform will be that the appropriate locations/guides are shown in a meaningful order and that inappropriate locations/guides are not shown. The Fail states are that inappropriate locations/guides are shown to the user, appropriate locations/guides are missing, locations/guides are shown in an arbitrary order, and inactive guides are shown to the user.

Server Latency

When testing server latency we first need to determine what the server's normal latency is when it isn't under load. After we determining the server's idle latency we can then create a load on the server by creating a script that makes several of API requests. While the server is handling all of the incoming API requests we can check how long the server takes to respond to request.

The pass condition for server latency is if the difference between the response times of the server underload and not underload are minimal. The fail condition is if the server response takes too long and jeopardizes the user's experience.

API Integration

When testing the api integration with our application we will create API requests with a user. We will check firebase to see if the request from that user reaches the server then we will check if the application gets the valid api response from the server. We are testing this way because this allows us to check if our all of our application's API post and get requests obtain a response from the server.

The passing criteria for API integration is if the application successfully sends and receives a response from the server. If the API response has usable information the API integration will have successfully passed our tests. The test will fail if there is no response from the server or if the response is invalid.

Profile Modifications

In the application, users will each have their own profiles where it may be modified. When testing profile modifications, we want to make sure that the changes the users are making are accurately being saved to the database. This is important because we want to have accurate recommendations based on the user's settings to encourage them to keep using the application.

There are a couple of things that are needed to meet the passing criteria. First, the profile modifications has to be accurately saved to the database in real time. If this step fails and the modifications are not being saved, we can not continue. If the first step passes, subsequent searches should be displaying results accordingly to the profile settings. If the results were to stay the same after the changes were made, it may fail the criteria (can depend on how drastic the changes were).

Report System

Our application focuses a lot on having users interact with each other. Not everyone online will get along with each other, and some may cause problems for another user. When a user has a problem with another user, we want to give the user the option to be able to report another user for reasons such as harassment, verbal abuse, hate speech, inappropriate behavior, and more.

We will have to test that it works correctly by checking and making sure that the reports being made are legitimate. If the report made by a user was valid and there is proof for it, it should bring the reported user's ratings down. Furthermore, the reported user should no longer appear on the guide list to the user making the report. Another goal in this testing is to make sure that the reports are not being fabricated to intentionally harm another user's ratings for the wrong reasons.

Deliverables

Deliverables that will come from our test plan will be the scripts that we used to perform our stress testings. Each test items and features may contain multiple scripts to ensure that the passing criteria is being met. Furthermore, we plan to also have demonstration videos of the testing features to show that they are in fact working as intended. These demonstration videos can also be used to highlight key features of our application to draw in a user base.

Responsibilities

The team members involved with each individual feature will also be responsible for the testing of that feature. This will allow us to streamline and improve our workflow by not requiring each team member to completely understand the goings-on of each feature behind the scenes. Compartmentalizing the features and their testing allows us to properly delegate and distribute

workload, allowing for shorter development time and ultimately a higher quality product at the finish.

Schedule for Planned Testing

The schedule for planned testing is based off of the completion of our deliverables. As deliverables approach their deadlines we will simultaneously test each deliverable. The deliverable schedule is determined by the importance of the function in our application. The order of the deliverables should be completed from the most critical to least critical functions with dependencies/prerequisites of critical functions being created and tested first.