Wait-Less Test Report Summary

Group 22: Omar Salas, Brian Goldenberg, Harsh Gupta, Emmanuel Utomi

The Wait-Less application is a mobile application designed to improve the efficiency of the restaurant. This is done by the task system where employees and managers can assign and complete tasks so they always know their objective as they go through a dinner service. Due to the complex system architecture, there are many aspects of the application that must be tested to verify and ensure that it's working as intended.

One major code snippet that was tested was the Future Builder in the frontend. This code is in charge of getting all the data for the completed tasks and ensuring that it's displayed in the application. There are several states that this component can have. The first state is retrieving the data. During this state there is a loading icon displayed so the user can know that tasks are being retrieved. When the data has been retrieved, there are two outcomes from there. One is that there are tasks and they will be displayed properly. The other is if there are no tasks, then a custom text is displayed. Knowing that this is a very important piece that connects the backend, frontend, and database together, it was integral that we tested it thoroughly to ensure every state is working as intended.

Another important piece of code that was tested was one of the widget for the popup dialog. This code is used all throughout the application to provide detailed description and it uses various data type variables as the input. The code is mainly focused on the UI part of the popup and has various requirements. The major requirements are that the latest Flutter SDK should be used and be provided with proper import statements. The main build function of a Flutter application must be executed. The proper BuildContext object is passed into the test function and then BuildContext successfully builds the Container widget, it also utilizes object fields to determine the shape, color, and border radius values of the Container's BoxDecoration object. We also tested that the created BoxDecoration widget contains a BoxShadow object of color black, blurRadius, and offset. Overall the test was important to compare various widgets and provide a better understanding to the developers on various devices of the application interface.

We also tested the endpoints for our API. These end points tests ensured the http requests are working and the entire API is processing data correctly. In order to ensure our API is working correctly, we made sure that the endpoint is calling the correct function. On top of this, we needed to make sure the database is properly modeling the changes the endpoint made. In order to do this after the endpoint was called, we went back to the database to ensure the change was successful. Finally, we needed to ensure the proper status code was being returned to the

client calling the API. We used junit tests to ensure that the proper status code was returned after the function was complete.

On a similar note, an important section of code is the authentication process for application login. This code block handles the transaction of encrypted data between the client and the backend servers. This process must be handled with integrity, in order to protect the sensitive data of users. Passwords are encrypted before being sent over the network. This process can return a few statuses. The successful state returns the status code 200 OK. This results in an active logon, and a returned copy of the current user's profile data for the client to store locally. In the case that incorrect credentials are provided, the backend will return various status codes depending on the circumstances. Erroneous status codes are returned for unknown username, incorrect password, as well as when an SQL database access error occurs.

For the code reviews, we used a general good coding practice checklist since our application is written in multiple languages. Using this checklist we were able to review each other's code for general good practice skills like commenting, variable meaningfulness, and comprehension.