CS 3110 MS3 Test Plan

Team Members: Flynn Kelleher (fpk24), Harsh Patel (hhp26), Ilia Raiszadeh (ipr6), Joonseok Jung (jj575), Kayla Runkel (kmr227)
Team Name: Pixel Princesses
PM: Allen Wu (axw4)
PG: Rachael Close (rlc355)

Description of approach to testing:
Our approach to testing the system was to primarily split up our functions into two distinct types which we would test differently. The first type of functions we had were logic functions that were in charge of the backend handling of the expenses data, their representation, and their manipulation. For these functions, we created a rigorous test suite to test all different types of inputs, corner cases, and attempted to get maximum line coverage (Note: bisect was not used). Because these functions actually return values instead of just unit, we were able to write concrete tests for them. We used a combination of black box and glass box tests to do this. For every function that it was applicable we tried to begin with black box tests to ensure that our output matched our expectations for a given input. From there, we tried to get coverage of the entire function using glass box tests. The second class of functions that we had were the frontend functions that handled the display of the different elements of the graphical user interface. Since these functions returned unit, we had to test them by trial and error where we would just run the program and navigate to the particular window to see how the frontend function behaved. The main things we were checking for was that the clickable areas and our objects matched in size and that the pages navigated to where they were supposed to go.

Anything we omitted testing:
As mentioned above, tests for our frontend portion of our program will not be visible in the test suite. This includes any function that displays a window, a text box, text, etc. However, many of these functions use other functions that have been tested. Given that we were unable to test some of these functions, we spent ample time running our program with different inputs and clicking different button combinations to ensure a clear user experience. We also did not test on inputs that our users are explicitly told not to input (like a string when asked for a number) because we can not guarantee to our users that things will continue to operate correctly if they do so.

Why test suite demonstrates correctness of system:
The test suite demonstrates correctness of the system because each testable function has been tested using a combination of black box and glass box tests. These tests have provided reasonable coverage across all behaviors to ensure that as many edge cases are covered as possible. Additionally, we tested using a variety of inputs, including zero values, empty lists, corner cases, and large values. The test suite

tests all of the functions that are in charge of delivering current outputs to the user based on the inputs. This paired with our manual tests of the frontend functions, ensures that we have tested the complete correctness of the logic of the system and the visual representation of it.