• Unit tests (25 points): Include a file/directory named ‘Testing’ in your Git Repository. There should be details (can be in a separate file in the directory) provided by each team member about the module and the functional testing they have done. Each team member picks a module or module and lists the equivalence classes and the test cases selected to cover all equivalence classes.

Team Name: SlugHub
Team Member: Lawrence Tam, Naum Markenzon, Mason Nguyen, Xuhua Feng, Zhelin Li

Unit Tests

Mason:

- Google Script to extract data from Google Calendar to Google Sheet:
    - Ran the script and compared the results to that of the Google Calendar
    - Adjusted the variables to get the desired data output
- Loading Screen:
    - Ran the app multiple times to see if the loading screen transition worked
    - Had to rewrite some code after realizing it made all the other screens transition to the home screen
    - Test imports for the unique functions that the Loading Screen calls

Zhelin:

- Python script to download .csv file from Google Sheet
    - Ran the script and see if the file download correctly
    - Updated the Google Sheet and ran again to see if the file been covered
- ScrollView:
    - Made sure it works and looks nice
- Import testing:
    - Floating button, ran pages contain floating button and try if they work

Lawrence:

- Import testing
    - Package import testing
        - Visual Studio Code helps with the testing process by lighting up imports when they are correct
    - Image Requires() input testing
        - Made sure all images file paths are called correctly
    - App.js pages import testing

        - Made sure all page imports were proper. Visual Studio Code helps with the testing process by lighting up imports when they are correct
    - Expo Bundler will return an error if any import or require() functions are called incorrectly. Since there are now no errors, we can be sure all of these are called correctly
- CSS styling formatting testing
    - Visual testing::
    - Debugging syntax:: Expo Bundler will return an error if any bugs are found in the code. I made sure that no CSS bugs are found that cause the application to crash

Xuhua:

1. Import testing:
    a. For the background and button picture, we use the require() function to make sure the directory or file does exist. Expo will show error messages during the bundling process
    b. For the packages and functions, VSCode will visually indicates whether it's being used or not so that we can get rid of unnecessary imports and unused functions.
2. Browser and link testing:
    a. For buttons that opens up a link using the built-in browser, I made sure that all the links are correct and tested them using the button in our app. Even if the website stopped functioning, the button will not be impacted since it will still open the browser, but just like any other browser, it will show a 404 not found message indicating that the website is not working.
3. CSS file testing
    a. For this part everything is rendered in real time and we can just debug it with our eyes and make sure everything looks the way we want it to.

Naum:

1. Filesharing code (file_share.py)
    a. Code will not work without given the correct parameters for host, port, username, password, source path, and destination path
2. Dictionary Generating Code (classes.py)
    a. In order for a class to be added to the dictionary, the page must have a title and header (which is not apparent in web pages without a valid course.) However, if there is an 404 HTTP Page Not Found error, my code still parses the dictionary, as that page STILL contains the title and header pages. If given more time, I would store the various .json dictionaries locally in case there is the 404 error. Despite this error, there should not be any problems with the code.

3. Dictionary generating code in JavaScript (cs_bs.js)
    a. The code that the JavaScript file uses for the fetch request works if the url exists (which it does in most cases), meaning that as stated above, 404 errors can go through as an error to the fetch request but yet still have an incorrect dictionary created. If the correct JSON dictionary is loaded, then there are no problems.
    b. Not an error per se, but the text for the courses wraps to the end of the line, which is not aesthetically pleasing as it blocks a portion of the picture.