Manuel Test Documentation

General Weekly Timeline of Testing:

September 28th – Given Project

September 30th – Developed the test cases as outlined in the initial project spec, envisioning the edge cases we would need to encounter and preparing for accounting for data storage and displaying user data. We had also developed some basic Android Studio test cases to assure our program could run on multiple platforms.

October 7th – Used the test development process to better understand what features we would need to implement in the report part of our project, focusing heavily on the screen time and notification listener.

October 14th – Using the testing process realized that we would need to store the data that kept on getting destroyed on every iteration of our application, and as a result began to look more into the specific surrounding onPause() and onSaveInstanceState(). Kinner realized that asking for permission would be necessary for the notification listener due to this interaction with testing.

October 21st – Worked on what test cases would be needed to assure that the device ran with multiple activities open, as we needed to transition within screens. Azhan and Vidy looked into UI testing (ensuring certain textboxes were constrained, and the Textboxes would be modular in regard to size). With this process we also realized we should try to graphical represent data and looked into that aspect of the UI.

October 28th – At this point the process became more manual as we were running multiple simulations of our program in an effort to understand exactly how the code was functioning and ensuring its passing our existing test cases. Also looked into how we would plan out both recommend and alert as those weren't implemented in time for the mentor check-in.

November 4th – We are mainly focusing on recommended and report with the feedback we have and are thinking about how exactly to simulate the messages for alert, although we are realizing that may not be possible.

Manual Tests Cases for features:

*Note: Screen Time/Pickups = STP

Report

1. STP saved when exited app

a. Purpose: To ensure that the values for STP were maintained even after the user has closed our app from running in the background.

b. User needs to open app, then exit app, pretend to use another app, and then reopen app and click the start/stop button. This generates ST. User than exits app, waits a period of time, and then reopens app. The same ST should be displayed as before until the start/stop button is pressed again.

c. No preconditions necessary, app must simply be running in the background.

2. Clearing Data upon button click

   a. Purpose: To ensure that the values that have been generated can be removed so that the user can record new screen times.

   b. User generates STP via process outlined in 1. User then clicks the clear button, and then exits app. When user re-enters the app, the app should display no STP at all, and it should be a blank slate.

   c. Precondition: There must be STP values already set for apps. The user can clear even if there is nothing there, except then the user will see no difference.

3. Data isn't overridden when relaunching app

   a. Purpose: To ensure that the previous STP value isn't the one which is changed when the app is launched again after exiting.

   b. User generates STP via process outlined in 1. User then exits the app and stops it from running in the background. User than launches app again, beginning screen timer automatically. User then exits app to use social media apps. User then goes back to app (which was in the background) and presses the start/stop button. The previous time should be present with the new STP right next to it.

   c. Precondition: There must be STP values already set for apps.

4. Data does get destroyed when emulator stops

   a. Purpose: To check that if the emulator stops, the app isn't somehow retaining data (sanity check)

b. User generates STP via process outlined in 1. User then exits app and turns off emulator. User than relaunches the emulator with the code, and runs app. No previous values should be present.

c. Precondition: There must be STP values already set for apps, so we can ensure they are gone after relaunch.

*Note Notification test cases were in Junit on GitHub

Recommend

1. Accurately finds top 5 worst apps

   a. Purpose: This is the basic test for recommend, the test that determines whether it is doing what it is intended for.

   b. The User goes to the Recommend section, and automatically, the UI should display the top 5 worst apps determined by the algorithmic method in the UI.

   c. Precondition: User has received notifications and has STP values in Report.

2. Accurately keeps the data across multiple attempts to recommend

   a. Purpose: To ensure that even across multiple recommends, it always considers the past values for that day.

   b. The user finds recommendations based upon prior usage. User exits app. The User uses apps that were at the bottom of the top 5 for a substantial period of time. The User goes to recommend again, the top 5 apps should be adjusted with the new activity of the user.

   c. Precondition: Prior recommendations must have been provided.

3. Doesn't recommend any app that's not on list

   a. Purpose: To ensure that Recommend isn't somehow accessing apps that weren't provided by Report (sanity check)

   b. The user goes to recommend and looks at the apps that are depicted as the top 5. They should be apps that the user has just used in the last iteration or on that day, and no other apps.

c. Precondition: A prior report should have been conducted, so that the data is stored in preferences.