Unit Testing Report

SocialShopper
KMS Inc.
12/01/2019

Add Friends Page:

- **Ralph:**
    - Tested to see if you can add the same friend multiple times. This was done by pressing the name of the friend that was wished to be added and analyzing the output.
    - Tested this page within the List setup and found it doesn't work (In the known bugs page). Tested by adding a friend while in the list set up and seeing if that friend actually appeared as an available participant.
    - Tested to see if the friend appears in the database after selecting a user. This was tested by adding a friend to the friends list in app and seeing if the friend was actually added to the friends list in the users section in the database.
- **Axel**:
    - Tested to see if both friends are linked to each other in the database.
- **Alan:**
    - Tested to make sure that you can't add a friend twice, or add yourself as a friend by manually adding users repeatedly to ensure consistent behavior.

App Settings Page:

- **Ralph:**
    - Tested to see if the switch list tile can save its state (at the time it couldn't but was later fixed). This was tested by changing the state of the switch and leaving the page. Upon returning I saw whether or not the state change remained.
    - Tested to see if the alert dialog will appear for the non-implemented features.This was tested by clicking on the non-implemented switches and seeing if the alert dialog stayed or not.
- **Axel**:
    - Tested to see if dark mode switch setting was saved after exiting app.
    - Tested whether or not changing switch state change the application's theme.

- **Alan:**
    - Tested to make sure the dark mode style was applied appropriately across the entire application by running through each control flow path possible and visually inspecting the UI to make sure it was implemented correctly.

Authentication File:

- **Axel:**
    - Tested whether Google Sign in actually created a user in the Firebase database.
    - Tested whether Google Sign in successfully created an account in Firebase Authentication.
    - Tested whether email and password sign in actually created a user in the Firebase database.
    - Tested whether email and password sign in successfully created an account in Firebase Authentication.
    - Tested whether update functions within this file actually updated database entries.
    - Tested whether new users required different update functions as to not overwrite certain data in the database.

Friends List Page:

- **Axel:**
    - Tested, using print statements, the user’s uid, in order to ensure correct user’s friends list was being retrieved.
- **Alan:**
    - Tested to make sure that friends are being deleted properly by creating a small code snippet to add and delete several friends and compare the result to the known value.

Globals File:

- **Ralph:**
    - Tested to see if the contents of the file remained consistent across the pages of the app. Tested Via Print statements.
- **Omar**
    - Tested if the data being retrieved for database was accurate by comparing to database
- **Alan:**
    - Tested to make sure variables are pulled correctly and consistently throughout all pages of the application by importing the globals.dart file into every page. Then, I made several reads and writes and compared the new values across pages to make sure they were all updated correctly.

Item Input File:

- **Omar**
  - Tested the manual entry text field for item name. At first i had it in a button that saved on ‘submit’, but that didn’t work since the user may not always click the checkmark. I ended up changing it to a ‘on changed’ button.
  - I did the same for the price and quantity text fields. I changed them to be a ‘on changed’ button. I also made the keyboard a numbers only keyboard since the variable that will hold that data will always be a number. I tested it if the user entered a letter the program would crash.
  - For the check marks that allows the user to click who is associated with the item. I only made sure that the user names were in the text field and nothing more. At first it was showing the userId, which shouldn’t be there.
  - Lastly, the save button. I tested by writing something for the item name, but not entering the price and quantity and i did select a user associated with the item. The program would then crash. The program did not like when there were empty variables. I then initialized the variable for item name to an empty string, the price variable to 0.0 and quantity to 0.0. I also put an empty list if the user didn’t checkmark a user. Therefore, none of the variables that would be populating the database would have null and the program wouldn’t crash.

List Setup Page:

- **Ralph:**
  - Tested to see if the user can leave all of the fields blank. Analyzed the outputs for leaving each of them blank.
  - Tested to see if the User ID was available and consistent across all processes within this page. I tested this through the use of print statements and analyzing the outputs of internal dependant functions.
  - Tested to see if multiple of the same participant is allowed to join a list. This was tested by attempting to add a user to a single multiple times and analyzing the outputs.
  - Tested that when a file is created in the list setup page that it also appears in the database for each of the users that are in that list. I did this by going to firebase and looking at each of the users data and seeing if that file is added to their lists array under users.
- **Alan:**
  - Tested to make sure that a new list is set up in the database with the correct default schema, making sure to eliminate null values to prevent UI errors.

- **Sean:**
    - Tested to see if we can add friends to the list setup page
    - Tested to make sure information of the list matched up with that in database

List View Page:

- **Axel:**
    - Tested the precision of output doubles.
    - Tested the removal of list from database in order to ensure that they were deleted in both the user’s list array and the list collection.
- **Omar**
    - Tested that items that get added to the list actually get displayed on screen and populate the database. I tested by checking database and checking app.
- **Alan:**
    - Tested to make sure that items are deleted from a list properly and that the changes are reflected in the database by creating and deleting items while observing the state of the database.
    - Tested to make sure that all of the group and individual totals were calculated correctly using print statements to validate the results of calculation.
    - Tested to make sure that a hint is displayed when no items are present in the list by creating empty lists, as well as adding and deleting items to lists to ensure consistent behavior.

Login Page:

- **Ralph:**
    - Tested to see if the user is allowed to enter an invalid character in the password.
    - Tested to see if the user is allowed to enter an invalid Email.
    - Tested to see if the user is allowed to enter an invalid length of characters.
    - The above were all done through analyzing the outputs from their respective text box entries.
- **Axel:**
    - Tested whether any password would work.
    - Tested whether user could use invalid email.
    - Tested whether log in could occur with empty text fields.
    - Tested whether Google Sign in button worked as intended.
    - Tested whether Sign Up button worked as intended.

Main File:

- **Axel:**
    - Tested whether provider was ancestor of receiving widget.
    - Tested whether build widget correctly routed to Login Page or Menu Page.

Menu Navigation File:

- **Ralph:**
    - Tested to see if the file navigates to the correct pages upon button press.
    - Tested to see if the User stays consistent throughout the app. I did this by embedding print statements that printed out the users name and ID in all 4 of the respective paths you can take from the menu page and analyzing the outputs.
- **Alan:**
    - Tested to make sure the profile page loads correctly with all of the user's information by logging in and testing the page on startup with several user accounts.
    - Tested to make sure lists are properly deleted from the database by adding and deleting several lists and observing the state of the database.

Mock Store Page:

- **Omar**
    - Tested if the correct data in the database was being displayed onto the app.
    - Tested if the correct data gets added to the user list when they add it to their list.
    - Tested if the user didn't enter a quantity the program would crash. Made it that if the user didn't enter a quantity it would be initialized to 1.
    - Tested if the user added more to an item they already have in the list that it would just add the quantity to that item and only display

Payment Page:

- **Axel:**
    - Tested whether page correctly displayed all available payment apps.
    - Tested whether page allowed the opening of other apps on the phone, given user gesture input.

Sign Up Page:

- **Ralph:**
    - Tested to see if the user is able to enter not enter anything in any field.
    - Tested to see if the user is able to enter an invalid email.
    - Tested to see if the user is able to enter an invalid password length.
    - Tested to see if the confirm password works in the case that the user enters a different password than the original.

  - The above were all done through analyzing the outputs from their respective text box entries.
- **Omar**
  - Tested if I could enter without entering values when signing up
  - Tested if i could have 1 character for a password
- Sean:
  - Tested to see if you can not fill out the name field when signing up
- **Axel**:
  - Tested to see if the user is able to enter not enter anything in any field.
  - Tested to see if the user is able to enter an invalid email.
  - Tested to see if the user is able to enter an invalid password length.
  - Tested to see if the confirm password works in the case that the user enters a different password than the original.
  - Tested to see if button correctly initiated the creation of an account in the database.

Store Select Page:

- **Ralph:**
  - Tested to see if the Store Name can be passed between files. Test was implemented through the use of print statements.
  - Tested to see if the non-implemented buttons created an alert dialog
- **Axel:**
  - Tested background color of page by changing its value.
  - Tested build widget in order to ensure that a MaterialApp was returned by this page.
- **Alan:**
  - Tested to make sure that the selected store’s name is passed back to the list setup page correctly by using print statements to show the return value from the store select page and compare it to the desired result.

Theme File:

- **Axel:**
  - Tested whether a ThemeNotifier object would actually notify listener on setTheme() call with a print statement.