Unit Testing | DataDiet | Peanuts Gang | December 2, 2019

Jeremy Manalo

- **Product Scanning**
    - Tested the scanning of various barcodes for successful and non-successful results by finding different barcodes and seeing if they would either return a message if the barcode was not found or displaying the product information

Ashley Cline

- **Scanner Settings / Personal Settings**
    - Tested the storing of settings changes to Firebase by switching diets on/off and adding/deleting allergies/intolerances in UI while watching the live updates to Firebase
        - EX: User is vegetarian so they switch that diet on and allergic to aspartame so they add that to the allergy list. Their PersonalSettings document in Firebase: Vegan: false, Vegetarian: true, Pescatarian: false, Kosher: false, Ketogenic: false, Paleolithic: false, Allergies: [“aspartame”]
        - EX: User scanned food for their friend who is allergic to buckwheat, peanuts, and tree nuts but now deletes them from their allergy list. Their ScannerSettings document in Firebase: Vegan: false, Vegetarian: false, Pescatarian: false, Kosher: false, Ketogenic: false, Paleolithic: false, Allergies: []
    - Tested the updating/preservation of settings in UI by closing/reopening the settings pages to ensure the settings do not disappear as well as manually changing the settings fields in Firebase and checking that the UI displays the changes when the settings are opened
        - EX: User is vegetarian so they switch that diet on and allergic to aspartame so they add that to the allergy list, then they close out of the settings and come back later. The settings still display vegetarian switched on and aspartame in the allergy list.
    - Tested the resetting functionality by tapping reset on a settings page with diets switched on and allergies in the list and observing the live changes to UI and Firebase.
        - EX: User scanned food for their friend who is pescatarian and allergic to buckwheat, peanuts, and tree nuts but now resets their scanner settings so they can reload it with their own dietary restrictions. Their ScannerSettings document in Firebase: Vegan: false, Vegetarian: false, Pescatarian: false, Kosher: false, Ketogenic: false, Paleolithic: false, Allergies: []. Their scanner settings UI has all diets switched off and an empty allergy list.

- **Editing Account Settings**
    - Tested the editing of the user's name by providing a new first and last name and watching the live changes in Firebase and the account settings UI. Tested the error checking of the edit name functionality by providing a blank first and/or last name and observing the error message.
        - EX: User wants to change their name to be just their first name, they delete the text in the last name text field, and press "Done". The error message "Both fields are required" is displayed and prevents change of name. User returns to account settings and the name displayed is their original name.
    - Tested the editing of the user's email by providing a new email and watching the live changes in Firebase and the account settings UI. Tested the error checking of the edit email functionality by providing a blank email and observing the error message.
    - Tested the editing of the user's username by providing a new user and watching the live changes in Firebase and the account settings UI. Tested the error checking of the edit username functionality by providing an invalid username as well as a taken username and observing the error message.
        - The criteria for a valid username are:
            - Between 3-28 characters long (1)
            - Only contains alphanumeric characters, underscores, or dots (2)
            - Underscores and dots can not be *next* to each other (3)
            - Underscores and dots can not be used multiple times in a row (4)
            - Username can't end or start with a dot or underscore (5)
        - The test inputs used to check if those criteria were met are:
            - For (1): "jeff" for a valid name and "ty" for an invalid name
            - For (2): "doggy.dog_w" for a valid name and "?-1" for an invalid name
            - For (3): "d_5.2" for a valid name and "d_.2" for an invalid name
            - For (4): "d.p.d" for a valid name and "d...p" for an invalid name
            - For (5): "a_pple" for a valid name and "_apple" for an invalid name
        - EX: User wants to change username to "eric" but that username already exists within the database, the error message "Username already exists" is displayed and the change is rejected.
- **History Page**
    - Tested the printing of product titles, scanned settings, and found settings by adding a bunch of fake data to the History document in Firebase and observing the behavior of the history page when it is opened.

        - EX: User scanned a bag of hot cheetos with their scanner settings including a vegetarian diet and cheese, milk, egg and whey allergies. Their history page displayed:
          Cheetos Flamin' Hot Crunchy Cheese Flavored Snacks
          Scanned: Vegetarian, cheese, milk, egg, whey
          Found: Vegetarian, cheddar cheese, buttermilk, whey, whey protein
    - Tested the deletion of individual products from history as well as the overall clearing of history by deleting/clearing products and watching the live changes to the UI and Firebase.

<u>Eric Zamora</u>

- **Finding/Adding Friends**
    - Testing for valid searches and valid friend relations:
        - The criteria for valid searches are:
            - The user can tap the search bar and type in the desired user's full name or username and they will successfully be retrieved and pop up on the table view.
        - The test inputs used to check if those criteria were met are:
            - If the user with the name "Jeremy Manalo" and username "jer" exists in the database, then the user will show up as a table cell as the characters in their name or username are typed into the search bar.
            - If the user with the name "Jeremy Manalo" and username "jer" exists in the database, if the user types in "jerEMY", then that user will still show up in the table view because the search is case-insensitive.
            - If the user with the name "Jeremy Manalo" and username "jer" doesn't exist in the database, then the user will not show up when being searched upon in the database.
            - If the user with the name "Jeremy Manalo" and username "jer" exists in the database, if a user types in "Jem", then the user will not show up because the search no longer satisfies the test criteria.
        - The criteria for valid friend relations are that one user finds a friend then sends a friend request which then sends a request to the other user. This becomes visible on firebase with the incoming friend's UID and on the "Pending Requests" scene. The other user then hits accept and then both users now have the other's UIDs on their friends list.

        - This was successfully tested using multiple simulators and different accounts that were able to add/accept each other and view each other as friends

- **Login/Sign Up**
    - Testing for valid and invalid account fields (username, passwords, email)
        - The criteria for a valid username are:
            - Between 3-28 characters long (1)
            - Only contains alphanumeric characters, underscores, or dots (2)
            - Underscores and dots can not be *next* to each other (3)
            - Underscores and dots can not be used multiple times in a row (4)
            - Username can't end or start with a dot or underscore (5)
        - The test inputs used to check if those criteria were met are:
            - For (1): "jeff" for a valid name and "ty" for an invalid name
            - For (2): "doggy.dog_w" for a valid name and "?-1" for an invalid name
            - For (3): "d_5.2" for a valid name and "d_.2" for an invalid name
            - For (4): "d.p.d" for a valid name and "d...p" for an invalid name
            - For (5): "a_pple" for a valid name and "_apple" for an invalid name
        - For all of these test cases, the valid inputs were successfully accepted and printed to the console with a success boolean and the invalid inputs tossed an invalid username error to the user until they provide a valid one.
        - The criteria for a valid password are:
            - Between 8-32 characters long (1)
            - Must contain at least one alpha and numeric character
            - Can contain special characters and combination of lower and uppercase
        - The test inputs used to check if those criteria were met are:
            - For (1): "four4444" for a valid password and "four444" for an invalid password
            - For (2): "bikinib0ttom" for a valid password and "bikinibottom" for an invalid password
            - For (3): "aPPle12?" for a valid password
        - For all of these test cases, the valid inputs were successfully accepted and printed to the console with a success boolean and the invalid inputs tossed an invalid password error to the user until they provide a valid one.
        - The criteria for a valid email are:
            - Must follow the format of a string local part + @ + string domain

- The test inputs used to check if those criteria were met are:
  - "john@smith.com" for a valid email and "john@@smith.com" for an invalid email
  - "john123@smith.com" for a valid email and "john#%&@smith.com" for an invalid email
  - "j@smith.com" for a valid email and "john.smith.com" for an invalid email
  - "j@s.org" for a valid email and "12345678901234567890123456789012345678901234567890123456789012345678901234xx@s.org" for an invalid email
- Firebase has their own criteria for valid emails, but the valid test inputs all successfully registered the user whereas the invalid ones gave an error saying it was unable to create the user.

Kenneth Mai

- **Import Settings**
  - Tested the importing of a friend's personal settings into current scanner by switching the desired friend in "Friend Settings Controller". Changes were reflected in Firebase and in the scanner of the user.

Example Test Cases:

- Selected friend named "Eric Zamora"
  - Returns personal settings of "Eric Zamora", which are loaded into the user's current scanner.

Eric's Personal Settings:	User's Current Scanner:
Allergies:	Allergies:
Oranges	Oranges
Ketogenic: true	Ketogenic: true
Kosher: false	Kosher: false
Paleolithic: false	Paleolithic: false
Pescatarian: false	Pescatarian: false
Vegan: true	Vegan: true
Vegetarian: false	Vegetarian: false

- Unselected friend named "Jeremy Manalo"
  - Returns personal settings of current user because no friend was selected. Settings are then loaded into the user's current scanner.

    Current User is "Kenneth Mai"

User's Personal Settings:	User's Current Scanner:
Allergies:	Allergies:
Peanuts	Peanuts
Apples	Apples
Ketogenic: true	Ketogenic: true
Kosher: true	Kosher: true
Paleolithic: true	Paleolithic: true
Pescatarian: true	Pescatarian: true
Vegan: false	Vegan: false
Vegetarian: false	Vegetarian: false

- **Share Personal Settings With Friends**
    - Tested by toggling the switch of desired friend. If toggled off, the friend should not be able to import the user's settings because they are not visible in their import scanner preferences. If toggled on, the friend should be able to select the user in their import scanner preferences.

Example Test Cases:

- User has shared with friend "Eric Zamora"
    - The user should now be visible in Eric Zamora's import scanner preferences.

- User has shared with friend "Jeremy Manalo"
    - The user should now be visible in Jeremy Manalo's import scanner preferences.

- User has unshared with friend "Jeremy Manalo"
    - The user should no longer be visible in Jeremy Manalo's import scanner preferences.