System and Unit Test Report
Life Reminders App
Off-By-One
Team Members: Jayden Navarro, John Gemignani, Alex Gonzalez, Kevin Cheng, Josh Innis
3/10/15

System Test Scenarios:

Sprint 1: Sprint 1 only had backend files so we used manual testing files to test these. We did not perform any system tests for Sprint 1.

Sprint 2:

- As a developer, I need to store reminders so that I can use reminders later. (Ability to be notified of reminder):
    1. Create a new reminder, and place it in Storage
        a. populate the fields with different values so that we know it is unique
    2. Retrieve reminder
    3. Reminder is recreated as it should be
- As a customer, I need to be able to see the list of reminders I have set and their notification settings:
    1. User is on All Notifications page
    2. All Notifications are displayed with their proper settings displayed as well
- As a customer, I want to be able to add my own reminders so that I can customize it the way I want:
    1. User presses plus button, populates the name of the reminder
        a. User can add a notification to it or choose to close the reminder
    2. User presses back button.
    3. Reminder is displayed with it's proper name in All Reminders indicating that it has been stored.
- As a customer, I need to be able to edit the information of a reminder so I can set each reminder the way I want
    1. User presses on a reminder
    2. Reminder page pops up and the user edits fields
        a. name = "Hello"
    3. The user presses back button
    4. All Reminders displays what was edited inside of the reminder

Sprint 3:
- As a developer, I want to push user created notifications to the notification bar
    1. Alarm goes off and the broadcast receiver gets it
    2. If the alarm is enabled and is in memory a notification is pushed

    3. A notification is displayed in the notification bar with the proper notification information

- As a customer, I want to add Reminders to individual Lifestyles or Notifications to individual Reminders
    1. The user presses on a reminder or a lifestyle.
    2. The user presses the plus button to add a new event to it
    3. The user edits the new event and presses the back button
        a. name = ”Test Event Name”
    4. The user returns to the reminder or lifestyle page they were originally in
    5. The user sees the new event in their list of children events, as it should
- As a developer, I want to be able to delete user created lifestyle/notification/reminder
    1. User long presses on the event, presses okay on dialog box
    2. All Events no longer displays the event indicating that it has been removed
- As a user, I want to modify the lifestyle/reminder/notification
    1. User presses on an event
    2. User edits fields in the edit page
        a. name = “Test Name”
    3. User presses back button
    4. In the All Event view the changed details are displayed
- As a customer, I want to be able to modify the settings
    1. User opens nav bar and selects settings
    2. Settings are modified
        a. Default can be switched to one of the three top switches
        b. Delete One Time Notifications is on or off
    3. The app responds appropriately to the different settings (all possibilities of settings were tested thoroughly)

Unit Tests:

Our unit tests are stored in the folder called “Tests” inside of our GitHub repository.

The full path is: Off-By-One/LifeReminders/app/src/main/java/com/jdndeveloper/lifereminders/Tests

And the link is:
https://github.com/JDNdeveloper/Off-By-One/tree/master/LifeReminders/app/src/main/java/com/jdndeveloper/lifereminders/Tests

NotificationTester.java (Jayden Navarro): Notification objects were created and alarms were set to test them. All functions were called and all types of input that would change the state were tested.

NotificationStorageTester.java (Jayden Navarro): This was used to test whether storage was properly storing Notificatins. Multiple notifications were created and had unique attributes setup for each. Then the notifications were committed to storage and retrieved to make sure that they matched the original objects.

SprintPresentationTester.java (Jayden Navarro): This was used to test the overall functional of the app at the end of Sprint 1. It sets up an alarm and pushes a temporary notification to the notification bar.

LifestyleTester.java (Kevin Cheng): Lifestyle objects are created as well as created and removed for assurance that removes also works. All functions are called and all types of input that would change the states are tested.

GsonTester.java (John Gemignani): Was designed as a proof of concept test for GSON, as a test of GSON functionality, and was used to generate test keys used throughout the application.

ReminderTester.java (Alex Gonzalez): This was used to test the core functionality of a Reminder. Running the first test tests to see if a reminder can properly add individual “keys” to a Reminder and then attempts to clear/clean the reminder. Test2 will attempt to add more than the max number of Notifications, returning with an error. Test3 is a test to see if a user wants to add a complete list to a Reminder, though this feature isn’t used, was still tested/

Since our app has a lot of GUI components, we didn’t have that many files that we could develop manual tests for. All other files were tested by using the app and trying to break it. Many bugs were discovered through this process and we found it to be a good way to learn more about our system. We used printout statements in our GUI code to figure out what was going on in different areas, so in a way the testing was actually done in the GUI files themselves.

Additional built in tests (John Gemignani, and others on the team): Our application has failure, test, and default key strings, return values, toasts and log messages to aid in tracking and debugging failures as well as to aid in testing. This process has been in place, in the application, since inception - through all 3 sprints. While they may not be obvious by having separate and distinct files they were used nonetheless. Additionally, many operations validate and do sanity tests on data manipulated.