Assignment Specifications

Keepfit

Table of Contents

Section I

Group Members

1. Zhenyi Yang zhenyiya@usc.edu 2239204516
2. Yi Xu yxu70832@usc.edu 2969873928
3. Gloria Liu liuglori@usc.edu 5512683227
4. Peidi Xie cindyxie@usc.edu 4927833566
5. Kevin Lee klee3663@usc.edu 8101223380

Section II

Preface

Keepfit continuous to the progress to the testing phase. Using black-box and white-box, we dig deeper into how sustainable Keepfit infrastructure and system is. As an iOS mobile application, there are still multiple aspects the team can improve upon regards to home-exercising.

With the “core” requirements tentatively done, it is an imperative step to conduct different tests on the Keepfit system. Once a user registers with Keepfit, the user should be able to conduct at-home workouts while not running into any security or infrastructure issues. Users should be able to navigate and utilize the mobile application to the fullest extent. With the capability of now creating accounts and saving personal information, it is imperative to ensure that the system is reliable for both user and developer as we continue on to develop the mobile application.

The document tests the system to see if existent or new bugs can be identified and fixed. In retrospect, we also try to diversify test cases as much as possible to cover different parts of the system.

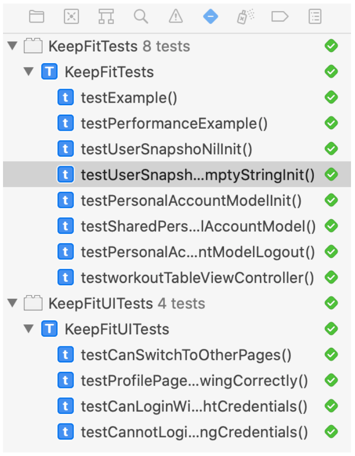
This testing document is intended for the developers of the Keepfit Project.
Current Version of This Document: 1.0. In this version, we specify the different testing conducted, both black-box and white-box, based on our previous core implementation. Throughout the documentation, readers will see multiple test cases for different parts of the Keepfit mobile application and how we identified and fixed the ubiquitous bugs.

Section III

Introduction

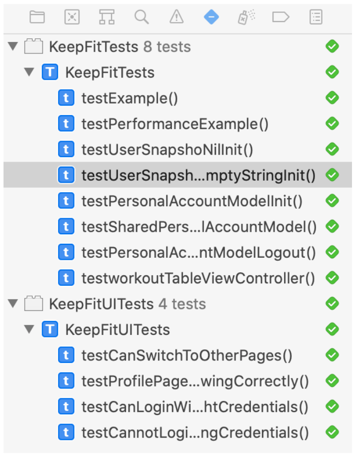
As Keepfit continues to grow in infrastructure, it is important to thoroughly test different parts of the foundation of the application. With different reliance on the components with each other, like firebase and personal information, ensuring these systems are tested with white-box and black-box benefits the current state of the application as well as the future.

All whitebox test cases are located in KeepfitTest/KeepFitTest.swift file within the repository and switch to “Testing Menu " and right clicking on the corresponding test/tests to run it/them, as shown below. .



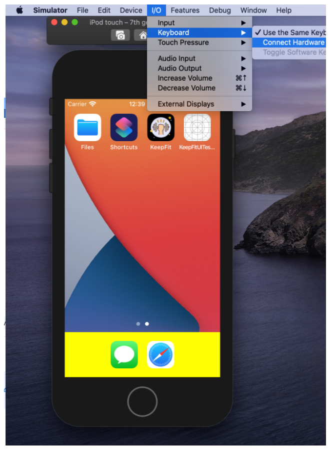
```
▼ KeepFitTests 8 tests                      ✓
  ▼ KeepFitTests                            ✓
      testExample()                         ✓
      testPerformanceExample()              ✓
      testUserSnapshoNilInit()              ✓
      testUserSnapsh...mptyStringInit()     ✓
      testPersonalAccountModelInit()        ✓
      testSharedPers...lAccountModel()      ✓
      testPersonalAc...ntModelLogout()      ✓
      testworkoutTableViewController()      ✓
▼ KeepFitUITests 4 tests                    ✓
  ▼ KeepFitUITests                          ✓
      testCanSwitchToOtherPages()           ✓
      testProfilePage...wingCorrectly()     ✓
      testCanLoginWi...htCredentials()      ✓
      testCannotLogi...ngCredentials()      ✓
```

All blackbox test cases are located in KeepfitUITests/KeepFitUITests.swift. You can run these test cases by switching to the “Testing Menu" and right clicking on the corresponding test/tests to run it/them, as shown below.



```
▼ KeepFitTests 8 tests                      ✓
  ▼ KeepFitTests                            ✓
      testExample()                         ✓
      testPerformanceExample()              ✓
      testUserSnapshoNilInit()              ✓
      testUserSnapsh...mptyStringInit()     ✓
      testPersonalAccountModelInit()        ✓
      testSharedPers...lAccountModel()      ✓
      testPersonalAc...ntModelLogout()      ✓
      testworkoutTableViewController()      ✓
▼ KeepFitUITests 4 tests                    ✓
  ▼ KeepFitUITests                          ✓
      testCanSwitchToOtherPages()           ✓
      testProfilePage...wingCorrectly()     ✓
      testCanLoginWi...htCredentials()      ✓
      testCannotLogi...ngCredentials()      ✓
```

Before running these cases, please make sure to configure the iPhone simulator. Go to I/O -> Keyboard and make sure that “connected hardware keyboard” is unchecked as shown below.



Section IV
Whitebox Testing

Test 1: testUserSnapshoNilInit()
Description: This tests makes sure that when UserSnapshot is initialized using nil arguments, it won't crash and it will be handled appropriately.
Result: The test runs successfully. No exception is thrown and all fields of UserSnapshot are initialized with nil.
Impact: This test did not uncover a bug.

Test 2: testUserSnapshotEmptyStringInit()
Description: This test makes sure that when User Snapshot is initialized using empty string arguments, it won't crash and it will be handled appropriately.
Result: The test runs successfully. No exception is thrown and all fields of UserSnapshot are initialized with an empty string.
Impact: This test did not uncover a bug.

Test 3: testPersonalAccountModelInit()
Description: Tests that the PersonalAccountModel can be initialized with no given argument.
Result: The test runs successfully. No exception is thrown.
Impact: This test did not uncover a bug.

Test 4: testSharedPersonalAccountModel()
Description: Tests the singleton pattern of the personal account model. Wherever change is made to the model, the change will be reflected in every other model in different references.
Result: The test runs successfully. The singleton pattern holds, where each access/change made to the model is dissipated to all references of the model.
Impact: This test did not uncover a bug.

Test 5: testPersonalAccountModelLogout()
Description: Tests that the PersonalAccountModel can successfully log out a user
Result: The test runs successfully. The PersonalAccountModel can successfully log out a user
Impact: This test did not uncover a bug.

Test 6: testworkoutTableViewController()
Description: Tests that the WorkoutTableViewController will correctly initialize and display the intended exercise options
Result: The test runs successfully. the WorkoutTableViewController correctly initialized and displayed the intended exercise options
Impact: This test did not uncover a bug.

Test 7: testExcerciseDetailConvertTimeToDisplay()
Description: Tests that the ExceriseDetailViewController can successfully convert integer values of time countdown to string display
Result: The test runs successfully. The controller is able to perform conversion.
Impact: This test did not uncover a bug.

Test 8: testStreamPageVCSearchEmptyResults()
Description: Tests that the StreamPageTableViewController can search using an empty string input without throwing errors
Result: The test runs successfully. The controller is able to perform the search without errors
Impact: This test did not uncover a bug.

Test 9: testLogin()
Description: Tests that the Firebase system can successfully login a user using the right credentials.
Result: The test runs successfully. The Firebase backend successfully authenticated the user.
Impact: This test did not uncover a bug.

Test 10: testLoginWithWrongPassword()
Description: Tests that the Firebase system will not log in a user with the wrong credentials.
Result: The test runs successfully. The Firebase backend successfully rejected the login of the user.
Impact: This test did not uncover a bug.

Test 11: testLogout()
Description: Tests that the Firebase system will successfully log out a user.
Result: The test runs successfully. The Firebase backend successfully logged out the user.
Impact: This test did not uncover a bug.

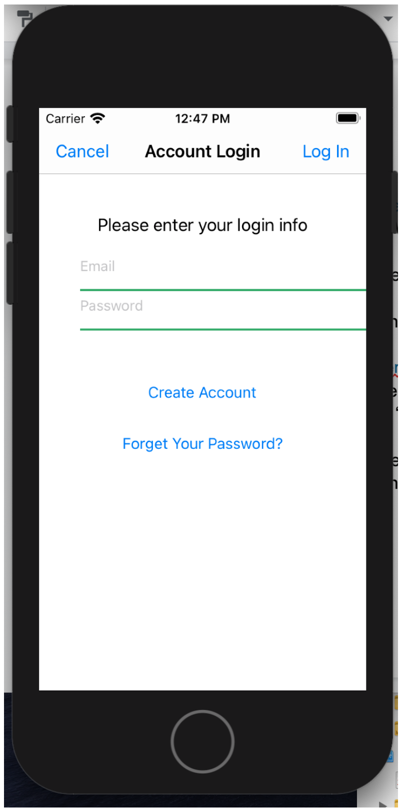
Test 12: testSwitchAccount()
Description: Tests that the Firebase system will let the user switch between accounts.
Result: The test runs successfully. The Firebase backend successfully lets the user switch from one account to another by logging out of an old account and logging back in with a new account.
Impact: This test did not uncover a bug.

Section V
Blackbox Testing

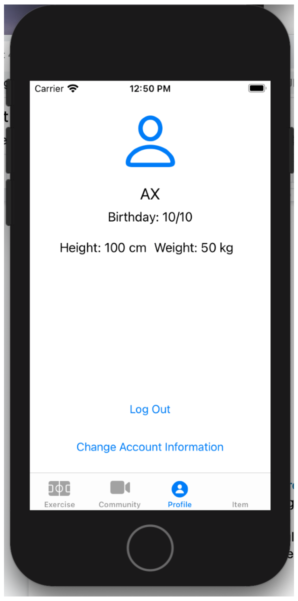
Test1: testCanSwitchToOtherPages()
Description: Tests that the WorkoutTableViewController will correctly initialize and display the intended exercise options

Result: The test runs successfully in the simulator. All tapping and text field actions are handled correctly.
Impact: This test did not uncover a bug.

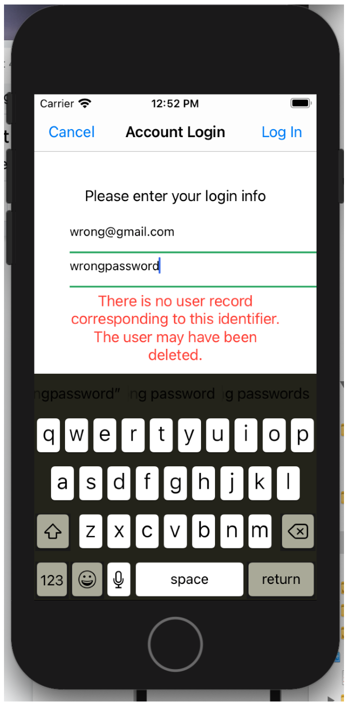
Test2: testProfilePageUIElementsShowingCorrectly()
Description: Tests that when we switch to the profile page while not logged in, a log in page will pop up and it will have the "signin button" as well as two textfields for the user to enter their account and password.
Result: The test runs successfully in the simulator as shown below.
Impact: This test did not uncover a bug.



Test 3: testCanLoginWithRightCredentials()
Description: Tests that in the login page, the user is able to enter their account and password information and log in using the correct credentials.
Result: The test runs successfully in the simulator as shown below.
Impact: This test did not uncover a bug.



Test 4: testCannotLoginWithWrongCredentials()
Description: Tests that in the login page, the user is able to enter their account and password information. However, when the wrong email/password combination is entered, an error message will appear and the user will not be able to login.
Result: The test runs successfully in the simulator as shown below.
Impact: This test did not uncover a bug.



There is no user record corresponding to this identifier. The user may have been deleted.

Test 5: testChangeAccountInfo()
Description: Tests that after logging in, the user is able to update his or her account information by clicking on the Change Account Information button. The user can change certain fields of information by typing texts into the fields, whereas for the fields not being typed information will be kept unchanged. At the end of the test case user account information will be restored to the version before the test case to ensure reusability of the test case.
Result: as the test case code specified, nickname and birthday info change successfully on the Profile page, whereas weight and height info don't change. The test runs successfully.
Impact: This test did not uncover a bug.

Test 6: testPasswordUpdateSuccess()
Description: Tests that after logging in, the user is able to update his or her password in the account configuration page. After the account configuration page displays "Update Success", the test case will log out and re-login with the new password to check if the password is updated successfully. At the end of the test case user account password will be restored to the version before the test case to ensure reusability of the test case.
Result: the password is updated successfully and the user is able to re-login with the new password. The test runs successfully.
Impact: This test did not uncover a bug.

Test 7: testPasswordUpdateFailed()
Description: Tests that after logging in, the user should not be able to update his or her password with invalid password format. The account configuration page should display an error message indicating the invalidity of the attempted password. The test case will then re-login with the original password to show that the badly-formatted password is not updated.
Result: three instances of badly-formatted password are not updated and the test case successfully re-login with the original password. The test runs successfully.
Impact: This test did not uncover a bug.

Test 8: testCreateUserUIElementsShowingCorrectly()
Description: Tests that in the Create Account page, the text entered in each text field is shown properly to the user. The test case will enter text for each text field and check if the value of the text field matches the corresponding text message.
Result: All text fields show correct messages. The test runs successfully.
Impact: This test did not uncover a bug.

Test 9: testCreateUserEmptyFields()
Description: Tests that in the Create Account page, a new user cannot be created if some of the text fields are left blank. In the test case all text fields are typed with correctly-formatted values except that the weight field is left blank. The Create Account page should display "Please fill in all fields" message.
Result: The Create Account page displays a "Please fill in all fields" message. The test runs successfully.
Impact: This test did not uncover a bug.

Test 10: testCreateUserInvalidPassword()
Description: Tests that in the Create Account page, a new user cannot be created if the password is badly-formatted. In the test case all text fields are typed with correctly-formatted values except that the password is not in correct format. The Create Account page should display "Please make sure your password is at least 8 characters, contains a special character and a number" message.
Result: The Create Account page displays a "Please make sure your password is at least 8 characters, contains a special character and a number" message. The test runs successfully.
Impact: This test did not uncover a bug.

Test 11: testCreateUserInvalidEmail()
Description: Tests that in the Create Account page, a new user cannot be created if the email is badly-formatted. In the test case all text fields are typed with correctly-formatted values except that the email is not in correct format. The Create Account page should display "Error creating user" message.
Result: The Create Account page displays "Error creating user" message. The test runs successfully.
Impact: This test did not uncover a bug.

Test 12: testCreateUserInvalidHeight()
Description: Tests that in the Create Account page, a new user cannot be created if the height is not a numerical value. In the test case all text fields are typed with correctly-formatted values except that the height is a string value. The Create Account page should display "Please fill in a number for height" message.
Result: The Create Account page displays a "Please fill in a number for height" message. The test runs successfully.
Impact: This test did not uncover a bug.

Test 13: testCreateUserInvalidWeight()
Description: Tests that in the Create Account page, a new user cannot be created if the weight is not a numerical value. In the test case all text fields are typed with correctly-formatted values except that the weight is a string value. The Create Account page should display "Please fill in a number for weight" message.
Result: The Create Account page displays a "Please fill in a number for weight" message. The test runs successfully.
Impact: This test did not uncover a bug.

Test 14: testPlayVideo()
Description: Tests that after switching to the Video page by clicking on the Item button on the menu bar and then clicking on the Video button, the user is able to click on a video and display it. The test case will display a video with the title “Dashanghuahuo” and no exception is thrown.
Result: the video “Dashanghuahuo” is successfully displayed with the video and audio fully functioning, and the test case doesn’t throw any exception. The test runs successfully.
Impact: This test did not uncover a bug.