

ParkHere Testing Document

November 10, 2016

Team Name: LazeeBear

Team 11b

Ryosuke
Mitsubayashi

Robert
Sieh

Zhicheng(Franklin)
Wang

Sally
Wei

4239440668

6502483653

6100373109

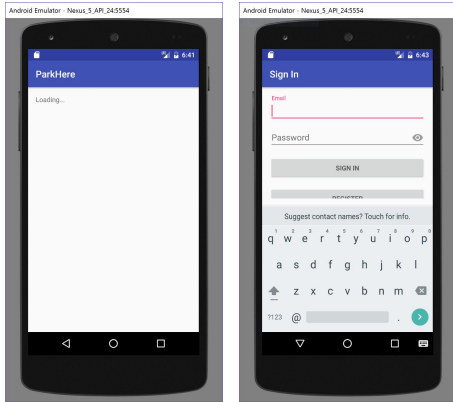
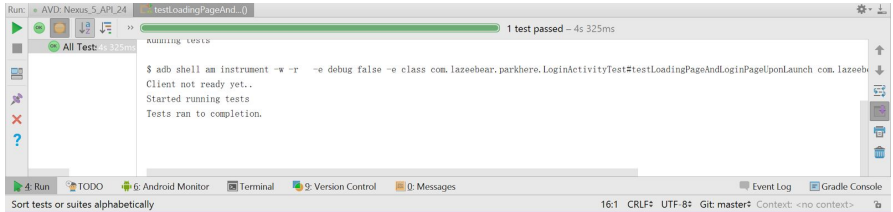
1145544502

Table of Contents

I. BLACK BOX TESTS	3
01 - TC_BB_01_Parkhere_Open>Loading_Page_and_Login_Page	3
02 - TC_BB_02_Parkhere_Login_Page_Register_Button	4
03 - TC_BB_02_Parkhere_Login_Page_Input_Verification	5
04 - TC_BB_03_Parkhere_Login_Page_User_Verification_Page	6
05 - TC_BB_04_Parkhere_Signup_Page	7
06 - TC_BB_05_Parkhere_Account_Page	8
07 - TC_BB_05_Parkhere_Account_Page_Owner	8
08 - TC_BB_05_Parkhere_Account_Page_Seeker	9
09 - TC_BB_07_Parkhere_History_Of_Spots_List_Owner	10
10 - TC_BB_07_Parkhere_History_Of_Spots_List_Seeker	10
11 - TC_BB_08_Parkhere_Spot_List_Edit_Post_Popup	11
12 - TC_BB_12_Parkhere_Spot_List_Edit_Spot	11
13 - TC_BB_13_Parkhere_Owner_Delete_Spot	12
14 - TC_BB_13_Parkhere_Reservation_Cancellation_Seeker	12
15 - TC_BB_10_Parkhere_Search_By_Address	13
II. WHITE BOX TESTING	14
01 - TC_WB_Server_Connector_Search	14
02 - TC_WB_Server_Connector_Signup	14
03 - TC_WB_Server_Connector_Sign_in	15
04 - TC_WB_Server_Test_Rate_User	15
05 - TC_WB_Server_Test_Modify_User	16
06 - TC_WB_Server_Test_Check_User	16
07 - TC_WB_Server_Test_Sign_in	17
08 - TC_WB_Server_Test_Sign_up	17
09 - TC_WB_Server_Test_View_User	18
10 - TC_WB_Server_Test_Search_Spot	19
11 - TC_WB_Server_Test_View_Spot	19
12 - TC_WB_Validation_Functions_Check_Valid_Email	20
13 - TC_WB_Server_Test_Post_Spot	20
14 - TC_WB_Server_Test_Delete_Spot	20
15 - TC_WB_	21

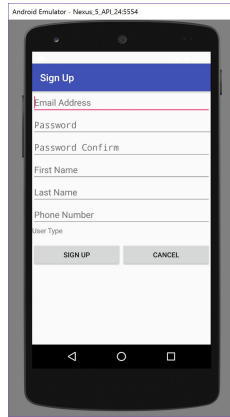
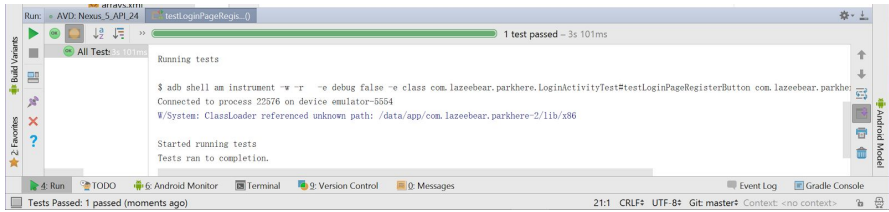
I. BLACK BOX TESTS

01 - TC_BB_01_Parkhere_Open_Loading_Page_and_Login_Page

Location of test case	src/androidTest/java/com/lazeebear/parkhere/LoginActivityTest.java testLoadingPageAndLoginPageUponLaunch()
Description of what the test case does / Directions to execute	<p>Tests that the app, after launching, ends on the login page.</p> <ol style="list-style-type: none">1. Open the app3. Wait.4. Check for existence of login page. <p>Expect the login page to show up.</p>
Rationale	<p>The loading page should show up when the application first launches. When the application is finished loading, the application should refresh to the login page.</p> <p>From the Parkhere Requirements Specification 5.3.1 “Loading page”</p> <p>5.3.1.a “Shown when app starts”</p> <p>5.3.1.b “The user is forwarded to the login page once loading is done”</p>
Result	<p>Success</p>  
Details	<p>Originally this test was meant to test the loading page, but due to the way Espresso works (testing the final state of the app after all actions have been executed), and that fact that the loading page</p>

	<p>disappears automatically, it is impossible to test the loading page directly with Espresso.</p> <p>So the loading page must be checked manually by a human.</p>
--	--

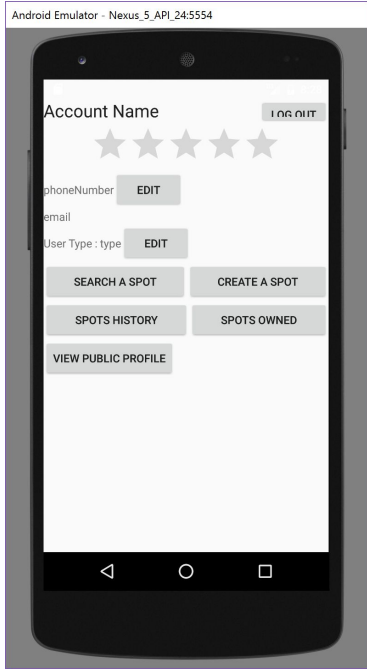
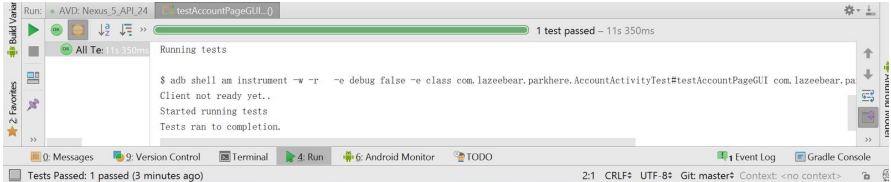
02 - TC_BB_02_Parkhere_Login_Page_Register_Button

Location of test case	src/androidTest/java/com/lazeebear/parkhere/LoginActivityTest.java testLoginPageRegisterButton()
Description / Directions	<p>Tests the completeness of the user interface of the login page and the components' functionalities.</p> <p>Prerequisites: 01 - TC_BB_01_Parkhere_Open_Loading_Page_and_Login_Page Assumes: You are at the login page.</p> <p>1. Click the "Register" button. <i>Expect the "Register" button to go to the "Sign Up" page.</i></p>
Rationale	<p>From the Parkhere Requirements Specification</p> <p>5.3.3 "Login page"</p> <p>5.3.3.d "New user button"</p> <p>5.3.3.d.i "When clicked the user is forwarded onto a user creation page"</p>
Result	<p>Success</p>  

Details	
---------	--

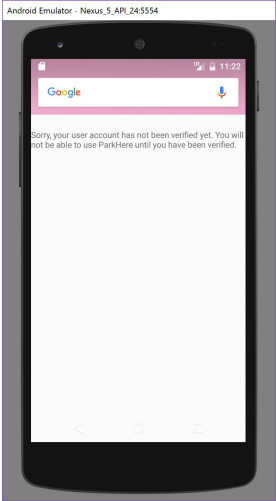
03 - TC_BB_02_Parkhere_Login_Page_Input_Verification

Location of test case	src/androidTest/java/com/lazeebear/parkhere/LoginActivityTest.java testLoginPageInputVerification()
Description / Directions	<p>Tests how the login page handles correct and incorrect username/password combinations.</p> <p>When a correct username and password are entered, the user should be sent to their Account Page.</p> <p>When an incorrect username and password are entered, the user should receive an error message and be unable to proceed.</p> <p>Assumes: You are at the login page. The Account Page works.</p> <p>1. Enter an existing username and the corresponding password. This pair belongs to a registered and verified user. (“rjason14@gmail.com“, “jerome“) <i>Expect the account page to show up.</i></p>
Rationale	<p>From the Parkhere Requirements Specification</p> <p>5.3.2 “Profile Page”</p> <p>5.3.2.c. “Login Button”</p> <p>5.3.2.c.i “When clicked the user is forwarded onto the Main page”</p> <p>The “Main page” of 5.3.2.c.i refers to this from the Parkhere Requirements Specification:</p> <p>5.3.7 “Profile Page (default page for owner)”</p> <p>Which was renamed “Account Page.”</p>
Result	Test passed.

	  <p>The account page that shows up when “rjason14@gmail.com” and “jerome” are used to log in has all the features that it needs to have on it.</p>
Details	The user rjason14@gmail.com lacks information updated.

04 - TC_BB_03_Parkhere_Login_Page_User_Verification_Page

Location of test case	src/androidTest/java/com/lazeebear/parkhere/LoginActivityTest.java testLoginPageUserVerificationPage();
Description / Directions	<p>Tests that a verification page pops up after an unverified user logs in.</p> <p>Prerequisites: TC_BB_02_Parkhere_Login_Page_Input_Verification</p> <p>Assumes: You are at the login page.</p> <p>1. Enter an existing username and the corresponding password. This pair belongs to an unverified user. (“abc@yahoo.com”, “Password!1 ”)</p>

	<p>2. Click the “login” button. <i>Expect a “verification needed” page to show up.</i></p>
Rationale	<p>From the Parkhere Requirements Specification 5.2.4.a.i “The user can only use the app after they have been verified”</p>
Result	<p>Passes. The correct verification screen shows up when the user is unverified, and the user is unable to proceed.</p> 
Details	

05 - TC_BB_04_Parkhere_Signup_Page

Location of test case	<p>src/androidTest/java/com/lazeebear/parkhere/SignupActivityTest.java testsSignUp()</p>
Description / Directions	<p>Tests the basic elements (email, password, user type) on the signup page and the ability to sign up correctly (be brought to the verification needed page).</p> <p>Assumes you are at the signup page.</p> <ol style="list-style-type: none"> 1. Type in an email that has not been used to signup with yet. (“abc1@yahoo.com”) 2. Type in a password. (“Password!1”) 3. Type in the same password into the confirm password box. (“Password!1”)

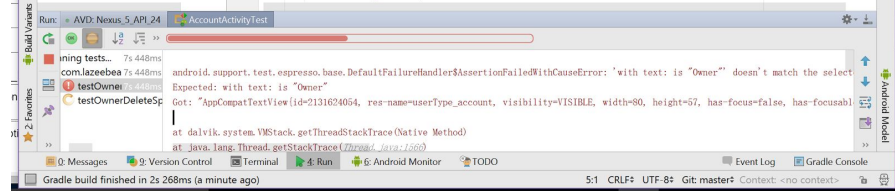
	4. Select Owner as the user type. 5. Click the "Sign up" button. <i>Expect to be brought to the Verification Needed page.</i>
Rationale	From the Parkhere Requirements Specification 5.1.2 "All users can be both owners and seekers. The user must specify whether he is an owner, seeker, or both upon signup." 5.2.4.a "A user account requires a first name, last name, email, and phone number" 5.3.3 "User Creation page" 5.3.3.f "Password" 5.3.3.g "Confirm password" 5.3.3.h "Owner/Seeker/Both dropdown" 5.3.3.i "'Complete' button"
Result	Fails. When a user first signs up, they will be unverified. Right now, the system directs them instead to the account page.
Details	

06 - TC_BB_05_Parkhere_Account_Page

Location of test case	src/androidTest/java/com/lazeebear/parkhere/AccountActivityTest.java testAccountPageGUI()
Description / Directions	Tests the existence of visible elements common to all user account pages. Assumes: You are at the account page. Account type (Owner, Seeker, or Both) does not matter, but the account has a type. 1. Check if this profile page is the user's own page. Only if yes, <i>Expects to see a "Settings" button.</i> <i>Expects to see a default profile picture.</i> <i>Expects to see user status (Owner, Seeker, or Both).</i> <i>Expects to see user rating (from 1 to 5 stars).</i>
Rationale	Previously known as the "Profile Page" in the Parkhere Requirements Specification. From the Parkhere Requirements Specification 5.3.7 "Profile Page" 5.3.7.a. "Shows default profile picture or one set by user explicitly" 5.3.7.c. "Shows listings of the individual"

	<p>5.3.7.d. “Shows user status”</p> <p>5.3.7.d.i. “Owner or Seeker or Both”</p> <p>5.3.7.e.i. “1-5 stars”</p> <p>5.3.7.g. “settings button”</p> <p>5.3.7.c and 5.3.7.d.i is tied too closely to account type, so although all accounts have those elements, they will be tested in the subsequent tests.</p> <p>TC_BB_04_Parkhere_Account_Page_Owner</p> <p>TC_BB_04_Parkhere_Account_Page_Seeker</p> <p>TC_BB_04_Parkhere_Account_Page_Both</p>
Result	Passes.
Details	

07 - TC_BB_05_Parkhere_Account_Page_Owner

Location of test case	src/androidTest/java/com/lazeebear/parkhere/AccountActivityTest.java testOwnerAccountPageGUI()
Description / Directions	<p>Tests the existence of visible elements unique to Owners.</p> <p>Assumes: The logged-in user is an Owner. The logged-in user has both user and spot ratings. The current page is the user’s own personal Account Page.</p> <p><i>Expects to see a “Create new post” button.</i> <i>Expects to see some ratings of users and spots.</i> <i>Expects to see a “Change user type” button.</i></p>
Rationale	<p>Note: Account page was previously known as the “Profile Page” in the Parkhere Requirements Specification.</p> <p>From the Parkhere Requirements Specification 5.3.7 “Profile Page” 5.3.7.e. “Shows ratings of user and parking lots (owner only)” 5.3.7.f. “Create new post button (owner only)”</p>
Result	<p>Fails. The logged-in user is not displayed on the page as an owner.</p> 

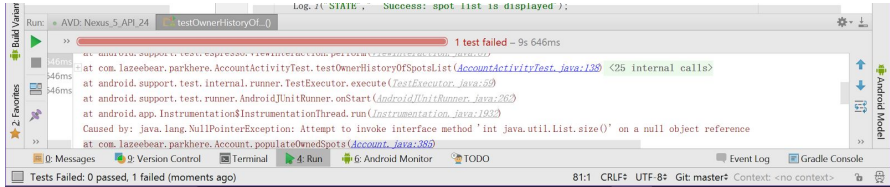
Details	
---------	--

08 - TC_BB_05_Parkhere_Account_Page_Seeker

Location of test case	src/androidTest/java/com/lazeebear/parkhere/AccountActivityTest.java testSeekerAccountPageGUI()
Description / Directions	<p>Tests the existence of visible elements unique to Seekers.</p> <p>Assumes: The logged-in user is a Seeker. The current page is the user's own personal Account Page.</p> <p><i>Expects to see a "View rentals" button.</i> <i>Expects to see a "Change user type" button.</i></p>
Rationale	<p>Note: Account page was previously known as the "Profile Page" in the Parkhere Requirements Specification.</p> <p>From the Parkhere Requirements Specification 5.3.7 "Profile Page" 5.3.7.i. "View rentals button (seeker only)"</p>
Result	Passes, but only if the user is only a seeker and not both a seeker and an owner.
Details	The current implementation has only one list for both "spots owned" and "spots history," so it cannot handle well a user that is both a seeker and an owner, who would normally have both lists.

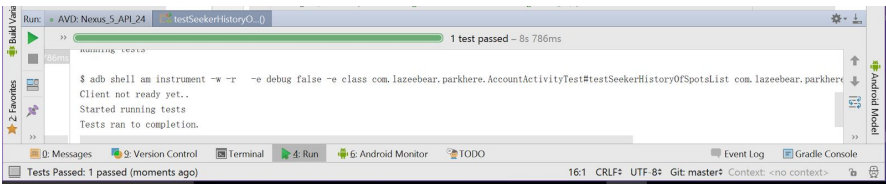
09 - TC_BB_07_Parkhere_History_Of_Spots_List_Owner

Location of test case	src/androidTest/java/com/lazeebear/parkhere/AccountActivityTest.java testOwnerHistoryOfSpotsList()
Description / Directions	<p>Tests that an Owner on their own Account Page will be able to see a list of all the spots they've posted after clicking a "Spots Owned" (view postings) button.</p> <p>Assumes: On the Account Page of the logged-in User The user is a Owner. The user has made posts.</p> <p>1. Click the "Spots Owned" button <i>Expects a list of the user's current postings to show up.</i></p>
Rationale	From the Parkhere Requirements Specification 5.1.12.b "Owners have a history of parking spots they've posted in

	<p>the past”</p> <p>5.3.7.h “View postings button (owner only)”</p> <p>5.3.7.h.i. “Forwarded onto a list of the current postings an owner has posted–each result shows the specifications set when created....”</p>
Result	<p>Fails. The list of posts is empty for the tested user.</p> 
Details	<p>The current implementation has only one list for both “spots owned” and “spots history,” so it cannot handle well a user that is both a seeker and an owner, who would normally have both lists.</p> <p>The user used to test this requirement does not have any spots to list.</p>

10 - TC_BB_07_Parkhere_History_Of_Spots_List_Seeker

Location of test case	src/androidTest/java/com/lazeebear/parkhere/AccountActivityTest.java testSeekerHistoryOfSpotsList()
Description / Directions	<p>Tests that a Seeker on their own Account Page will be able to see a list of all the spots they’ve booked after clicking a “view rentals” button.</p> <p>Assumes: On the Account Page of the logged-in User The user is a Seeker.</p> <p>1. Click “Spots History” button <i>Expects a list of the user’s rentals to show up.</i></p>
Rationale	<p>From the Parkhere Requirements Specification 5.1.12.a “Seekers have a history of parking spots that they’ve booked in the past”</p> <p>From the Parkhere Requirements Specification 5.3.7.i “View rentals button (seeker only)” 5.3.7.i.i “When clicked forwarded onto a list of the current rentals of the seeker”</p>
Result	Passes.

	
Details	

11 - TC_BB_08_Parkhere_Spot_List_Edit_Post_Popup

Location of test case	src/androidTest/java/com/lazeebear/parkhere/AccountActivityTest.java testOwnerHistoryOfSpotsListPopup()
Description / Directions	<p>If the user has access to their history of posted spots (is an Owner or Both), clicking on a spot should bring them to a page that will allow them to edit the information of the spot, to delete the spot, and to save the edits they've made.</p> <p>Requires: TC_BB_08_Parkhere_History_Of_Spots_Lists_Owner</p> <ol style="list-style-type: none"> 1. Click "Spots Owned" button. 2. Click on an Entry in the list. <p><i>Expect to see an "Edit" price button.</i> <i>Expect to see a "Delete" spot button.</i></p>
Rationale	5.7.3.h.i "...Clicking an entry will forward the renter to the 'create new posting' page, only now it will have a button to delete the posting and the submit button will say 'edit'."
Result	Passes. There exists an edit button next to Price and a delete button.
Details	

12 - TC_BB_12_Parkhere_Spot_List_Edit_Spot

Location of test case	src/androidTest/java/com/lazeebear/parkhere/AccountActivityTest.java tesOwnerHistoryOfSpotsListPopup()
Description / Directions	<p>Tests that an Owner can edit and update an owned spot's information.</p> <ol style="list-style-type: none"> 1. Logs in as an Owner 2. On the account page, click "history of spots" button 3. Click on a spot

	4. Click "Edit" button next to Price label. 5. Change the price 6. Click "Confirm" button. <i>Expect the information to have been changed.</i>
Rationale	Editing a posted spot (as an Owner or Both) should update the spot's information.
Result	(Cannot be tested on an empty list of history of spots.)
Details	

13 - TC_BB_13_Parkhere_Owner_Delete_Spot

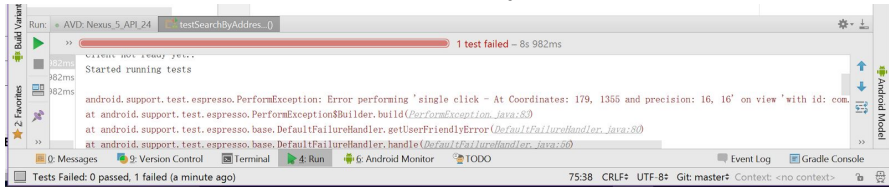
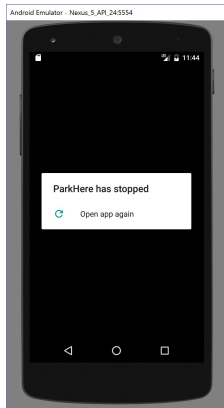
Location of test case	src/androidTest/java/com/lazeebear/parkhere/AccountActivityTest.java testOwnerDeleteSpot()
Description / Directions	Tests that an Owner can delete their own spot, and that the user is returned to the Account page afterwards. 1. Click "Spots History" button. 2. Click an Entry in the list. 3. Click "Delete" button. <i>Expect to be returned to the Account page.</i>
Rationale	From the Parkhere Requirements Specification: 5.3.h "View Posting Button (owner only)" 5.3.h.i "...it will have a button to delete the posting..."
Result	Fails.
Details	Hangs after clicking the button. The method to delete a spot was never connected properly to the server, so it will call the ServerConnector and hang while waiting for a response. The user will then never be returned to the Account page.

14 - TC_BB_13_Parkhere_Reservation_Cancellation_Seeker

Location of test case	src/androidTest/java/com/lazeebear/parkhere/AccountActivityTest.java testSeekerViewCancellationPolicy()
Description / Directions	Tests that a Seeker can view the spot they've rented and choose to cancel the reservation and see the total rental period. 1. Click "Spots History" button 2. Click an Entry in the list. <i>Expect to see a cancellation policy.</i>

Rationale	From the ParkHere Requirements Specification: 5.3.7.i “when an item is clicked the description of the rental is shown such as parking lot pictures, owner information, and rental duration, as well as a button for cancellation”
Result	Passes.
Details	

15 - TC_BB_10_Parkhere_Search_By_Address

Location of test case	src/androidTest/java/com/lazeebear/parkhere/SearchActivityTest.java testSearchByAddress()
Description / Directions	<p>Tests that a User can search by an address.</p> <ol style="list-style-type: none"> 1. Navigate to the Search page. 2. Type in an address on the search bar. (“707 West 28th Street, Los Angeles, CA 90007”) 3. Click “Search.” <p><i>Expect (at least one) result to show up.</i></p>
Rationale	From the ParkHere Requirements Specification 5.1.6 “Users can search for a parking space by locality to a specific address, date, and time slot.”
Result	<p>Fails when auto-run by Espresso.</p>  <p>Fails to receive a response when run manually.</p> 
Details	When run by Espresso, the test cannot find the “Search” button to click, possibly because the button’s location is changed (moved

	<p>downwards) when the address is filled in due to the length of the address.</p> <p>When run manually, the app freezes if it cannot connect to the Server.</p>
--	---

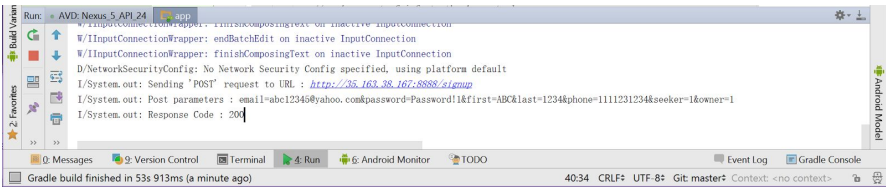
II. WHITE BOX TESTING

01 - TC_WB_Server_Connector_Search


Location of test case	src/test/java/com/lazeebear/parkhere/ServerUserFunctionalityTest.java search()
Description / Directions	<p>This test will check the search() functionality of the ServerConnector.</p> <p>Search() first signs in a user, then sets the expected returned variable of the search (a List<SpotDAO> object) to null. It then performs a search, after which it checks if the variable is not null.</p> <ol style="list-style-type: none">1. Perform a search.2. Check that the SpotDAO object is not empty.
Rationale	The basic connection between the server and the clients must be tested. Each method call to the server must be tested separately, so these formed our cases.
Result	Passes
Details	

02 - TC_WB_Server_Connector_Signup

Location of test case	src/test/java/com/lazeebear/parkhere/ServerUserFunctionalityTest.java signup()
Description / Directions	<p>This test will check the signup() functionality of the ServerConnector.</p> <p>Signup() signs in a user through the ServerConnector, then checks if the returned value from the ServerConnector is equal to "200," the success code.</p> <ol style="list-style-type: none">1. Perform a signup.2. Expect the success code to be "200."
Rationale	The basic connection between the server and the clients must be tested. Each method call to the server must be tested separately, so these formed our cases.

Result	 <p>Passes.</p>
Details	

03 - TC_WB_Server_Connector_Sign_in

Location of test case	src/test/java/com/lazeebear/parkhere/ServerUserFunctionalityTest.java signin()
Description / Directions	<p>This test will check the signin() functionality of the ServerConnector.</p> <p>Signin() attempts to sign in a user through the ServerConnector, then checks if the returned boolean is true (the sign in was successful).</p> <ol style="list-style-type: none"> 1. On the sign in page, type in a username and the associated password. ("abc@yahoo.com", "Password!1") 2. Expect the response code to be "200."
Rationale	The basic connection between the server and the clients must be tested. Each method call to the server must be tested separately, so these formed our cases.
Result	 <p>Passes.</p>
Details	

04 - TC_WB_Server_Test_Rate_User

Location of test case	Backend/testClient.py test_rate_user(self)
Description / Directions	<p>Tests if a user can be rated.</p> <ol style="list-style-type: none"> 1. Sign in ("rob@rob.com", "Password1\$"). Try to rate a user "rate@user.com" 2. Assert that the initial rating is 0. 3. Try to rate the user "rate@user.com" with a 5.

	4. Check that the rating of " rate@user.com " is now a 5.
Rationale	A user should be able to be rated. The server must be able to handle this. By writing separate tests for each point at which the rating is supposed to be handled, we can check for the different points of failure in the system and narrow down bugs.
Result	Success.
Details	<p>Discovered a bug in arithmetic to calculate new rating. Casting to float and int when necessary solved the bug.</p> <p>Note: test relies on changing a user's rating, so if running multiple times, you must update assertEquals</p>

05 - TC_WB_Server_Test_Modify_User

Location of test case	Backend/testClient.py test_modify_user(self)
Description / Directions	<p>Tests if a user's information can be modified (specifically, if their phonenummer and type of user can be modified).</p> <ol style="list-style-type: none"> 1. Sign in ("rob@rob.com", "Password1\$") 2. Check that the user's current phone number is "123-456-7890". 3. Check that the user is currently a seeker. 4. Try to change the user's phone number to "123-456-1111" and make the user not a seeker. 5. Check that the user's phone number is now "123-456-1111". 6. Check that the user is no longer a Seeker.
Rationale	A user must be able to sign in, and this functionality must be tested at the Server to differentiate its bugs from an other bugs on the frontend or connectors.
Result	Passes.
Details	<p>Discovered a bug where self.get_current_user (code written by a former teammate) does not work. Replaced with self.get_secure_cookie("user")</p> <p>Note: test relies on changing a user's information, so if running multiple times, you must update assertEquals</p>

06 - TC_WB_Server_Test_Check_User

Location of test case	Backend/testClient.py
-----------------------	-----------------------

	test_check_user(self)
Description / Directions	<p>Tests if the signed in user is the same as the user that attempted to sign in.</p> <ol style="list-style-type: none"> 1. Attempt to sign in. ("rob@rob.com", "Password1\$") 2. Check if the user is the same as the signed in user ("rob@rob.com") and not the same as other users ("rob1@rob.com", "rob45@rob.com").
Rationale	There needs to be a check for user equivalence since ParkHere will need to treat accounts that are not the logged-in user differently (won't display phone number, options to edit posts, etc).
Result	Passes.
Details	

07 - TC_WB_Server_Test_Sign_in

Location of test case	Backend/testClient.py test_signin(self)
Description / Directions	<p>Tests if a user with a certain email and password combination can attempt to sign in. Checks incorrect email and incorrect password.</p> <ol style="list-style-type: none"> 1. Takes a username and password 2. Attempts to sign in.
Rationale	The server must be able to handle requests to sign in with a certain username and password combination.
Result	Passes.
Details	Found an issue with salting the password, which did not work for emails not in the database. Fixed this by checking if getSalt could not find a salt, the parent function would immediately return false

08 - TC_WB_Server_Test_Sign_up

Location of test case	Backend/testClient.py test_signup(self)
Description / Directions	<p>Tests if a user that signs up with certain information will have that information created and saved on the server's side. Test that a user cannot sign up with the same email as a preexisting user.</p> <ol style="list-style-type: none"> 1. Attempt to sign up with a username and password.

	<p>(rob5@rob.com”, “Password1\$”)</p> <p>2. Check that the returned code is not a failure code.</p> <p>2. Attempt to sign up with another username and password that already exists. (rob@rob.com”, “Password1\$”)</p> <p>3. Check that the returned code is a failure code.</p>
Rationale	<p>A user should be able to sign up, and there should be a check in place in case the user tries to sign up with a username (email) that is already in use.</p> <p>The username acts as a primary key for each user account, and it is important that they are kept unique.</p> <p>This information must be checked on the server, which has access to the database.</p>
Result	Passes.
Details	Note: to run this test, you have to change the successful sign up info

09 - TC_WB_Server_Test_View_User

Location of test case	Backend/testClient.py test_view_user(self)
Description / Directions	<p>Tests that the server returns the proper user data for a user.</p> <ol style="list-style-type: none"> 1. Sign in as a user. (rob@rob.com”, “Password1\$”) 2. Attempt to view the user’s data. 3. Expect the user (rob@rob.com”) to have a rating of 0. 4. Expect the user (rob@rob.com”) to have a last name of “last”. 5. Expect the user (rob@rob.com”) to have a first name of “first”. 6. Expect the user (rob@rob.com”) to be a Seeker. 7. Expect the user (rob@rob.com”) to be an Owner. 8. Expect the user (rob@rob.com”) to have the email rob@rob.com 9. Expect the user (rob@rob.com”) to have a phone number of 123-456-7890.
Rationale	The user “ rob@rob.com ” was created ahead of time as a default user that is both a seeker and an owner, and with known data.
Result	Passes.
Details	

10 - TC_WB_Server_Test_Search_Spot


Location of test case	Backend/testClient.py test_search_spot(self)
Description / Directions	<p>Test that the server can search a spot by proximity to address.</p> <ol style="list-style-type: none"> 1. Sign in as a user ("rob@rob.com", "Password!\$") 2. Search an address. ("2801 Menlo Ave, Los Angeles CA, 90007") 3. Check that the returned object has address "2801 Menlo Ave, Los Angeles CA, 90007") 4. Check that the returned object has start date/time of "2016-11-12 12:00:00." 5. Check that the returned object has end date/time of "2016-11-12-14:00:00."
Rationale	"2801 Menlo Ave" is a spot that was created ahead of time with default values.
Result	Passes.
Details	

11 - TC_WB_Server_Test_View_Spot

Location of test case	Backend/testClient.py test_view_spot(self)
Description / Directions	<p>Tests that the server (http client) will return the correct spot information when trying to view a spot.</p> <ol style="list-style-type: none"> 1. Sign in as a user ("rob@rob.com", "Password!\$") 2. Search an address. ("706 West 28th Street, Los Angeles CA, 90007") 3. Load the spot data. 4. Expect the address to be "706 West 28th Street, Los Angeles CA, 90007." 5. Expect the type to be 0. 6. Expect it to not be covered. 7. Expect it to have the first type of cancellation policy. 8. Expect the price to be 10.00 9. Expect the start date/time to be "2016-10-12 12:00:00." 10. Expect the end date/time to be "2016-10-12 14:00:00." 11. Expect it to not be recurring.
Rationale	"706 West 38th Street" is a spot that was created ahead of time with known values.

Result	Passes.
Details	

12 - TC_WB_Validation_Functions_Check_Valid_Email

Location of test case	src/test/java/com/lazeebear/parkhere/ValidationsFunctionsTest.java validEmailAddress()
Description / Directions	Tests the validation function for checking a valid email address. 1. Check that an empty string is not valid. ("") 2. Check that just a nonempty string is not valid. ("yahoo") 3. Check that a string with only a ".com" is not valid. (".com")
Rationale	Tested that the email validation function checked for a nonempty string, for a ".com", for an "@", and that a valid address has to be all three. For full branch coverage, we tested for an empty string, a nonempty string without a ".com" or "@", a string with only an "@", a string with only a ".com", and finally a string with both an "@" and a ".com".
Result	Passes. 
Details	

13 - TC_WB_Server_Test_Post_Spot

Location of test case	Backend/testClient.py test_post_spot(self)
Description / Directions	Tests that the server responds with a success code when posting a spot. Opens a new http client and posts a spot
Rationale	Tests the spot posting functionality of the server
Result	Passes.
Details	

14 - TC_WB_Server_Test_Delete_Spot

Location of test case	Backend/testClient.py test_delete_spot(self)
Description / Directions	Tests that the server responds with a success code when deleting a spot. Opens a new http client and posts a spot before deleting it.
Rationale	Tests the delete functionality
Result	Passes
Details	

15 - TC_WB_Server_View_Postings

Location of test case	Backend/testClient.py test_view_postings(self)
Description / Directions	Tests that the Server returns a list of postings posted by an owner. Checks the first 2 spotIDs to confirm they are the same ones (ordered by history of posting) that the owner has posted
Rationale	Checks to see
Result	success
Details	Found an issue with one of the mySQL commands (index was out of bounds which prevented the command from working)