Unit Tests

Profile

ID	PT1
Description	Login with existing user credentials
Setup	Use FalseData.sql
Instructions	Set Login with the email and password, “DaveBurgess@gmail.com” and “password” Run login() on the above credentials Assert that the returned user object’s ID equals the ID of the user DaveBurgess@gmail.com
Expected Result	Return user object

ID	PT2
Description	Create a new account successfully
Setup	Use FalseData.sql
Instructions	Login using “newemail@gmail.com”, “password” as credentials Build a user using the above login info, and name input as “J”, “Unit”, “Test” Assert that the built user has the name “JUnitTest”
Expected Result	Return user_id over 1

ID	PT3
Description	Create a Group
Setup	Use FalseData.sql
Instructions	Login using "newemail@gmail.com","password" as the credentials Run createGroup() with the ID 1, name of “tempGroup”, and description “test”. Assert that the first member of “tempGroup” is the user ID with “newemail@gmail.com” Assert that the group with ID of 1 has the name “tempGroup”
Expected Result	Return group ID

ID	PT4
Description	Edit Account with a New Email
Setup	Use FalseData.sql
Instructions	Login to a user using “newemail@gmail.com” and “password” as the email and password and save the user ID Run editAccount() with the new primary email “newestemail@gmail.com” Assert that the same user ID has the primary email “newestemail@gmail.com”
Expected Result	Return true

ID	PT5
Description	Edit Account with New Email But Already Exists
Setup	Use FalseData.sql
Instructions	Login to a user using “newemail@gmail.com” and “password” as the email and password and save the user ID Run editAccount() with the new primary email “newestemail@gmail.com” Assert the result as “This email is already in use”
Expected Result	Return false

ID	PT6
Description	Edit Account Password
Setup	Use FalseData.sql
Instructions	Login to a user using “newemail@gmail.com” and “password” as the email and password and save the user ID Run editPW() with the new password: “newpassword” Assert by logging in with the email and new password that the returned user ID is the same
Expected Result	Return true

ID	PT7
Description	Find User
Setup	Use FalseData.sql
Instructions	Run finduser() on the ID of 2 and store it Assert that the returned user ID is not null
Expected Result	Return user object

ID	PT8
Description	Find Group
Setup	Use FalseData.sql
Instructions	Create a group with the name “testtest” and ID 1 Run findGroup() on the ID 1 Assert that the returned group has the name “testtest”
Expected Result	Return user object

ID	PT9
Description	List Users with an Empty List
Setup	Use FalseData.sql
Instructions	Run listUsers() searching on the name “blahblahblah” and store the returned user list Assert that the user list is empty
Expected Result	Return empty list

ID	PT10
Description	List Users
Setup	Use FalseData.sql
Instructions	Run listUsers() searching on the name “Dave” and store the returned user list Assert that the user list has the user ID of “Dave”
Expected Result	Return list of users

ID	PT11
Description	Add Friend
Setup	Use FalseData.sql
Instructions	Login to a user using “Daveburgess@gmail.com” and “password” as the credentials Add a friend named “Burger King “using addFriend() Assert that Burger King has a pending invite with the user Dave
Expected Result	Return true

ID	PT12
Description	Accept Friend
Setup	Use FalseData.sql
Instructions	Login to a user using “Daveburgess@gmail.com” and “password” as the credentials Add a friend named “Burger King “using addFriend() Assert that Burger King has a pending invite with the user Dave Accept the friend request using acceptFriend() Assert that Dave has a friend named Burger King
Expected Result	Return frue

ID	PT13
Description	Remove Friend
Setup	Use FalseData.sql
Instructions	Login to a user using “Daveburgess@gmail.com” and “password” as the credentials Remove a friend named “Burger King “using removeFriend() Assert that Dave does not have a friend named “Burger King”
Expected Result	Return true

ID	PT14
Description	Deactivate
Setup	Use FalseData.sql
Instructions	Login to a user using “Daveburgess@gmail.com” and “password” as the credentials Run deactivate() with the same input credentials Assert that the user ID tied to Daveburgess@gmail.com is hidden
Expected Result	Return true

ID	PT15
Description	Reactivate
Setup	Use FalseData.sql
Instructions	Login to a user using “Daveburgess@gmail.com” and “password” as the credentials Assert that the user ID tied to Daveburgess@gmail.com is no longer hidden
Expected Result	

ID	PT16
Description	Get Notifications List
Setup	Use FalseData.sql
Instructions	Login to a user using “Daveburgess@gmail.com” and “password” as the credentials Add a friend named “Burger King “using addFriend() Run getNotifications for Burger King to retrieve a list of notifications Assert that this list has a pending invite from Dave Burgess for Burger King to be friends
Expected Result	Return list of user ID’s

ID	PT17
Description	Facebook OAuth Access
Setup	Use FalseData.sql
Instructions	Using the credentials “facebookkloudbook@gmail.com” and “password”, receive an access token for that user’s facebook information Assert that the returned information has email facebookkloudbook@gmail.com
Expected Result	Return String of information

ID	PPT1
Description	Uploading a picture
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Selects Franz as the user Upload an image by the name of ABC123.jpg as Franz's profile picture Assert that Franz has a picture by the name of ABC123.jpg Run endDatabaseTest() to delete the database
Expected Result	Return true

ID	PPT2
Description	Deleting a picture
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Select the user Franz Delete the image ABC123.jpg from Franz Assert that Franz has no picture Run endDatabaseTest() to delete the datbase
Expected Result	Return true

Group

ID	GT1
Description	Adding Member
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Creates a group named "AddGroupMemberTestGroup" Selects that group and adds a member with the user ID of 6, Tommy Asserts that there exists a group member with the right user ID Run endGroupTest() to delete database
Expected Result	Returns the user ID of the user invited

ID	GT2
Description	Confirming a Member is of a group
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Create a group named “ConfirmMemberTestGroup” Add a member by the user ID of 6, Tommy, into the group Confirm Tommy as a member of the group Assert that Tommy is an active member of the group Run endGroupTest() to delete database
Expected Result	Return true

ID	GT3
Description	Removing a Member of a group
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Select the group kbGroup2 Grab the user Margaret and remove her Assert that the group does not contain a member named Margaret Run endGroupTest() to delete database
Expected Result	Returns a user ID

ID	GT4
Description	Setting a group Description
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Create a group with the name “AddSetDescriptionTestGroup” with the initial description “tests set description” Set a new description “newDescription” for the group Assert that the group’s description is “newDescription” Run endGroupTest() to delete database
Expected Result	Return true

ID	GT5

Description	Setting a group Name
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Create a group named “AddSetGroupNameTestGroup” Change the group name to “newName” by calling setGroupName() Assert that the group’s name is “newName” Run endGroupTest() to delete database
Expected Result	Return true

ID	GT6
Description	Setting Shared Information Level
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Create a group named “AddSetSharedInfoTestGroup” Set the shared Info level of that group to 2 Assert that the returned Shared Info level is 2 Run endGroupTest() to delete database
Expected Result	Return true

ID	GT7
Description	Set group as Active status
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Create a group named “AddSetGroupNameTestGroup” Set the active status of this group as false Assert that this group is inactive Run endGroupTest() to delete database
Expected Result	Return true

ID	GT8

Description	Changing a Member's priority level
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Create a new group named "AddSetChangeMemberPriority" Add a new member named Margaret with priority 1 Call changeMemberPriority() using admin on Margaret to change her priority level to 2 Assert that Margaret has a priority level of 2
Expected Result	Return true

ID	GT9
Description	Deleting a group
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Creates a group named DeleteTestGroup Asserts that this group exists via search Delete that group Asserts through searching that DeleteTestGroup does not exist
Expected Result	Return true

ID	GT10
Description	Getting a list of Active Members
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Select the group kbGroup3 Grab kbGroup3's active members Assert that the size of kbGroup3 is 3
Expected Result	Returns an array list of Group Members

ID	GT11

Description	Create a Group
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Create a group named “BOOOOOK” Search through the entire database for groups named “BOOOOOK” Asserts that there exists a group with the name of “BOOOOOK” Run endGroupTest() to delete the database
Expected Result	Returns a group ID

ID	GT12
Description	Finds a connection group
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Generate an ID by creating a group with the name “Group1” Find a group in the database matching that ID Assert that the ID exists in the database Run endGroupTest() to delete the database
Expected Result	Returns a group part of a profile

Search

ID	ST1
Description	Basic Search
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Select “Javier Bardem” as the user Performs a search for “Tom Cruise” and returns a linked list of ID’s Asserts if there is a list of ID’s, and that “Tom” is in it Run endDatabaseTest() to delete database
Expected Result	Returns a linked list of ID’s

ID	ST2

Description	Basic Search of Null
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Performs a search of “” Asserts if there is a list of ID’s, and that the list is null Run endDatabaseTest() to delete database
Expected Result	Returns an empty linked list

ID	ST3
Description	Advanced Search by Name
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Performs an advanced search on the name of “Franz Ferdinand Jorge Paco” and outputs a list of ID’s Asserts that “Franz” is in the list Run endDatabaseTest() to delete database
Expected Result	Returns a linked list of ID’s matching name query

ID	ST4
Description	Advanced Search of Null
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Performs an advanced search on null input Asserts that the returned list is null Run endDatabaseTest() to delete database
Expected Result	Returns an empty linked list

ID	ST5

Description	Advanced Search by High School
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Performs an advanced search using “Holyoke High” on the field of high school Asserts that “Franz” and “Tom” are correctly in the returned list Run endDatabaseTest() to delete database
Expected Result	Returns a linked list of ID’s matching high school

ID	ST6
Description	Advanced Search by Email
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Performs an advanced search of “test@gmail.com” on email field Asserts that “Franz” is in the returned list of ID’s Run endDatabaseTest() to delete database
Expected Result	Returns a linked list of ID’s matching email

ID	ST7
Description	Advanced Search by College
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Performs an advanced search of “Hampshire College” on the college field Asserts the returned list if “Franz” is in it Run endDatabaseTest() to delete database
Expected Result	Returns a linked list of ID’s matching email

ID	ST8

Description	Advanced Search by Company
Setup	Use dummyinsert.sql by startDatabaseTest()
Instructions	Performs an advanced search of “Amazon” on the company field Asserts that “Franz” is in the list of returned ID’s Run endDatabaseTest() to delete database
Expected Result	Returns a linked list of ID’s matching email

ID	ST9
Description	Advanced Search by Current City
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Performs an advanced search of “Amherst” on the current city field Asserts that “Franz” and “Javier” are in the list of returned ID’s Run endDatabaseTest() to delete database
Expected Result	Returns a linked list of ID’s matching email

ID	ST10
Description	Advanced Search by Hometown
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Performs an advanced search of “New York” on the hometown field Asserts that “Javier” is in the list of returned ID’s Run endDatabaseTest() to delete database
Expected Result	Returns a linked list of ID’s matching email

ID	ST11

Description	Advanced Search with all fields joined
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Performs an advanced search on “Holyoke High”, "Franz Ferdinand Jorge Paco", "Amherst", "Amazon”, “Hampshire College”, "test@gmail.com" on the fields of high school, name, current city, hometown, company, college, and email address respectively Asserts that “Franz” is in the list of returned ID’s Run endDatabaseTest() to delete database
Expected Result	Returns a linked list of ID’s matching email

ID	ST12
Description	Basic Search of Group
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Performs a basic search of “kbGroup” Asserts that the returned list of group ID’s has “kbGroup1” and “kbGroup2” Run endDatabaseTest() to delete database
Expected Result	Returns a linked list of ID’s matching group name

ID	ST13
Description	Basic Search of Friends List
Setup	Load dummyinsert.sql by startDatabaseTest()
Instructions	Perform a basic search for Javier Bardem Assert that Javier is in the returned results Perform a basic search Tom Cruise Assert that Tom is in the returned results Assert that the returned results list is size 1, since the friends list has only one Tom
Expected Result	Returns a linked list of ID’s matching name