Testing

Primary Author: Justin Renneke

User Acceptance Test Scenarios:

User acceptance testing is done by the end user to ensure all desired functionality is there and working the way the client actually wants it to work vs how the software developers think it should work. This is typically done as the last stage after other types of testing are complete and before shipment of the end product.

1. Test the outlined use cases by providing the end user a list of the use cases and any objects required to complete the tasks (for example, access to a data set for the Upload Data Set use case).
2. This testing environment should emulate real world usage conditions.
3. The flow of the system should be tested from start to finish. Can the user create a new account, log in, upload a data set, generate or upload a manifest for the new data set, search the database to find the uploaded data set via browse manifest, view it, download it, and log out when done without any issues?
4. Test the outlined use cases by providing the end user a list of the use cases and any objects required to complete the tasks (for example, access to a data set for the Upload Data Set use case).
5. The following use cases should be tested:
    1. Browse Manifest
    2. Contribute to Existing Dataset
    3. Download Info
    4. Generate Upload Manifest
    5. Search on Manifest
    6. Upload Data Set

Unit Test Scenarios:

The testing of single methods or functions during the coding process to validate functionality.

1. Early database establishment unit tests to verify that insert, update, and delete database actions are working correctly before continuing on to further system development.
   Failure cases: An insert, update, or delete statement is executed but a follow-up select statement does not return the expected result.
2. Login()
   Login with good info should flag the user as logged in and create the corresponding session variable

Login with bad info should initiate a notification of rejected login
Failure cases: username and password/password hash that aren't in database are allowed to log in; good username/password are not logged in; user gets logged in but has incorrect permission level

3. CreateNewUser()
Ensure invalid parameters are handled correctly (i.e. things such as username too long, etc)
Ensure that new user information is input into proper database fields
Verify username matches password in database
Failure cases: username is too long(>64 characters); sql insert statement is executed to create a new user in the database, but a select statement attempting to return matching info fails to find it; a search of the database fails to find a matching username and password pair
4. DeleteUser()
Ensure invalid parameters are handled correctly (i.e. user not in database, etc)
Search database for user after execution to ensure user information is deleted
Failure cases: an attempt is made to delete a user and that user cannot be found in the database, but the function indicates a successful deletion anyway; a database search for a user that should have been deleted returns the user's info indicating the user wasn't actually deleted
5. UploadDataSet()
Bad datasets or bad links to datasets should be rejected
Check database to ensure dataset is correctly inserted into proper database field
Failure cases: A dataset of size 0; a non-functional link to a dataset; A dataset that has supposedly been inserted into the database cannot be found there with a search of the database
6. GenerateUploadManifest()
If a manifest is to be uploaded, ensure invalid file types are not accepted.
If a manifest is to be generated, ensure function checks that all required data is input before creating the manifest.
Verify that the generated manifest returned by function is valid.
Failure cases: A user tries to upload a non-JSON type file; required field of manifest is left blank; generated manifest does not include all fields from form
7. List to be expanded as development continues...

Regression Testing:

Periodic testing done whenever changes have been made to the system to validate pre-existing functionality is still there. In other words, test everything that was already there to make sure we didn't break anything after adding new things to the system.

1. Ensure login still works as expected
2. Ensure non-authorized user can't access any restricted pages by going directly to the corresponding URL
3. Test all links to ensure they lead to the correct URL
*Ensure all pages led to by these links load correctly
4. Test all Forms

Check field default values Ensure only correct input type is accepted (i.e if a field is meant to accept a zip code, then it should only accept numbers)
*Verify user input is stored and used by the system

5. Ensure database integrity

Integration Testing:

Test interaction between different units of the system that must communicate with each other. Make sure these components are interacting correctly.
When to perform: Before the end of each sprint.

1. All links within the application should go to the correct page.
2. Anytime the application is given functionality to interact with the database, the integration of these components should be tested. Correctness of update, insert, delete, etc into the database should be verified and the proper display of any information retrieved from the database should be verified.
3. The log in system should be integrated across all web pages of the application. I.e. the logged in status of a user should persist across all web pages via a session variable until the session is ended and this should be tested for anytime a new page is added to the system.
4. Anytime information is passed between web pages, it should be verified that this information is correctly received and displayed.

Verification vs Validation:

Verification answers the question, ‘Are we building the system in the correct way according to the requirements and design specifications?’ Is the system well-engineered and error free? This is done with unit testing, integration testing, and regression testing.

Validation answers the question, ‘Are we building the right product to satisfy the customer’s needs?’ This is done with user acceptance testing.