Lab 11 - Testing Instructions

1. The test plans should include specific test cases (user acceptance test cases) that describe the data and the user activity that will be executed in order to verify the proper functionality of the feature.
2. The test plans should include a description of the test data that will be used to test the feature.
3. The test plans should include a description of the test environment that will be used to test the feature.
4. The test plans should include a description of the test results that will be used to test the feature.
5. The test plan should include information about the user acceptance testers. Please note that the deployed application can only be tested on the CU Boulder campus. So make sure to select your testers accordingly.

Create a branch in git
Git status
Git pull main
Git branch issue#4 (example for if you issue is labeled 4)
Git checkout issue#4 (switches you to new branch)

Done with Branch
Git add .
Git commit -m '<insert commit tag here>'
Git push —set -upstream origin issue#4
Git checkout main
Git pull origin main
Git checkout issue#4
Git merge main (merges main onto issue4)
Create a pull request on the github website

Feature 1: Login

- Test cases:
    - Test that a correct username and password actually logs you in and redirects to home page, as well as displaying user data ie profile picture, name, friends, etc
        - Will need to test SQL query works correctly?
        - Will need to ensure that you can type into the form and submit it appropriately
        - {username: lillian,

password: mypassword}
    - Test that an incorrect username and/or password DOES NOT log you in and keeps you on login page
        - Also gives feedback on error, e.g. “Incorrect username/password”
- Test data: dummy username/password, dummy user accounts with friends, profile pics, name, etc
    - Friends: Luke, Vikki, Justin, Freddy
    - Link to photograph of Lillian
- Test environment: development environment - cloud will give us details later on how many cores, ram, etc.
- Test results: Load the mixify page with the user’s profile picture at the top if successful, stay on login page if not and display an error message.
    - Assert(200, resultofApiCall)
- User acceptance testers: CU students - Every member of the lab group

Feature 2: Register

- Test cases:
    - Test that not filling out all required fields does not register the user in the database
      {username: lillian,
      password: mypassword}
      If password was missing it wouldn’t work - need to check that there is a username and password.
    - Test that once registered a user can immediately log in (use the same Login testing just immediately after Register is done)
    - Test that an already registered user can’t register again with the same username/email
- Test data: dummy usernames/passwords, existing accounts with the same username,
- Test environment: dev environment
- Test results: Load the mixify page and display a registration successful message if everything is inserted correctly, stay on the registration page and display an error message if not
    - Assert(200, resultOfAPICall)
- User acceptance testers: CU students, specifically other students in our recitation & all of us

Feature 3: Add & Remove Friend

- Test Cases

- Test that clicking the “Add” button next to a username correctly adds that user to the friends list in the database and on the front end - appears in their friends list on the main page
- The data that should be received is a code 200 if the friend was successfully added to the database
- Button type submit for clicking on the button and submitting on the form
- Test that removing a friend properly removes them from a user’s profile in the database, on the front end, and returns their username to the prospects list

* The test environment that will be used to test the feature is the website itself, checking that the friends list displays correctly on the front page, and by connecting to the database through docker & using postgres to query in the command line.
* Test results: friend successfully added to database for both the adder and the addee
    - assert(200, resultofAddtoDatabaseApiCall)
* User acceptance testers: CU students, particularly those in our recitation