

Features to be tested:

- **Testing login page -**

We will add a few dummy login accepted usernames and passwords into our tables to make sure it logs in and redirects to the correct page. We can manually insert this data while running our website but we will also create a python method that will have a password and username that should work and one that does not to make sure it redirects to the register page. Python is an easy language to use to test this feature through inputting our data and using asserts to execute the code.

TEST CASES:

1. Test Environment Connection: Postman
 - a. Create new collection in postman called lab10
 - b. Create new post request named "post_login"
 - c. Type: "http://localhost:3000/login" into request URL box and click send
 - d. Will show you contents of login page if successful along with status 200
 - e. Will show message `Error: connect ECONNREFUSED 127.0.0.1:3000` if unable to connect to login page.
2. Test that is supposed to succeed and redirects to the correct page
 - a. Test page connection
 - b. Test data input
3. Test that is supposed to fail and redirects to register
4. Test that makes sure a password is needed to login
5. Test that makes sure a username is needed to login

Data:

```
{'Email': 'jela4948@colorado.edu'  
'Username': 'jela4948'  
'Password': 'Jensen:'}
```

- **Testing Register page -**

We will add a few usernames and passwords into the database and verify if the program checks for duplicate accounts, validity of inputs, and connection status. We can manually insert this data into the page fields while running our website, through postman, and we will also create a python method that will check data validity, and if inputs are already in use, successfully redirect to the login page.

TEST CASES:

1. Test for valid email format
 - a. Example of Valid Email: {'jela4948@colorado.edu'}
2. Test for duplicate username and email via validity concept in code

3. Test for successful redirection to Login Page - via "Sign Up" button and "Login" button
 - a. If successful will connect to login page and show via browser
 - b. If not successful, the error message will be displayed in the console section of the inspect feature. (TBD)
4. Purposeful fail tests for email format
 - a. Example of NonValid Email: {'jela4948'}
 - b. Example of NonValid Email: {'@colorado.edu'}
 - c. Example of NonValid Email: {'jela4948@colorado'}
 - d. Looks for '@', 'email "username"', and domain name
5. Purposeful fail tests for duplicate credentials via validity concept in code
6. Test Environment Connection: Postman
 - a. Create new collection in postman called lab10
 - b. Create new get request named getRegisterPage
 - c. Type: "http://localhost:3000/register" into request URL box and click send
 - d. Will show you contents of register page if successful along with status 200
 - e. Will show message `Error: connect ECONNREFUSED 127.0.0.1:3000` if unable to connect to register page.

● Posting reviews -

This test is designed to determine if reviews which are being entered by the user is being put into the database correctly.

TEST CASES:

1. Test Environment Connection: Postman
 - a. Create new collection in postman called lab10
 - b. Create new post request named post_review
 - c. Type: "http://localhost:3000/reviews" into request URL box and click send a json object

Open a terminal, run the following command to look into the database:

```
docker compose exec db psql -U postgres
\c users_db
SELECT * FROM reviews;
```
 - d. See if the reviews table has the necessary data.
 - e. Test environment, sending requests from the browser
 - i. Replicate the steps from the postman test, but with the browser.
2. Description of test data.
 - a. A raw json object with text fields for {username, review}
3. Examples of the specific test data:


```
{
  "username": "Manas Gupta",
```

```
"review": "The food was not worth spending a penny over. Never going  
back!",  
"rating": 0  
},
```

4. Results
A result succeeds if the database entry has the same value(s) as the json object that was sent by either postman or the browser.
5. End user testing
 - a. The end user test will involve sending review data from the browser. This aspect of the test also tests front end elements.