Team Members:

- Owen Smith: osmith-CU
- Jingqi Liu(JanesyLiu): jili3523
- Mathew Kim: maki9472
- Patrick Taylor: pata0278
- John Hodgen: joho8797

Title: Pet-Gen

User Acceptance Testing:

- To incorporate unit testing into the project we can run test cases to return values from the SQL database at even the smallest attribute. This is important as the database is stored separately from the javascript code that is querying the database and we want an accurate working system. This test can be conducted by creating test cases to return values with a single feature, and a combination of features. This section of testing will also require the testing of the temp javascript array as we want to check the values stored in them each time to make sure they are random enough for the generator to be functional. Specific UAT:
    - If a user answers a question on the quiz that trait should be properly added to the user's temp class
    - The temp class that is filled in each question should properly store values after each question and the user should be able to see their answers at the end of the quiz
- To test the login features of our product, we can run cases in which a user might input a certain line of text in order to log in to the website. We have to ensure that the user login is functional and directs the user to the name generating program successfully without any issues. The test could be conducted by setting up cases in which the program ensures that the user has selected a username, password, and has input their email address in order to create an account associated with the website. The objective of this test would be to prove that the login features are functional and working as intended. The criteria for this test will be as follows: the user cannot input the same email address, the user cannot input an empty email address, and the user can successfully access the website upon the creation of an account. The test environment will most likely be a javascript function that ensures that no duplicates can be found, inputting the same email address twice to check, as well as inputting nothing to ensure that the user cannot access the website. We can ensure that the program is working when Mocha and Chai display that the program returned successfully. Specific UAT:
    - A user who enters the same email address should not be able to generate a new account
    - A user who enters nothing should not be able to generate a new account

    - Upon successful creation of a new account, the user should be able to seamlessly access the website
- To test the randomization process of our product, we can run our randomization function using the whole name dataset and tally the number of times each name is rendered. If this value relative to the total number of tests shows randomness across time, we can definitely say that our pseudo-randomization algorithm works as intended. The objective of this test would be to prove that our randomization algorithm does not yield any particular name sets more frequently or infrequently than it should. The criteria for this test will be a predefined threshold for each name. If a name occurs +/-10% more frequently than it should occur statistically, we can say that this name's choice odds are being inflated to a statistically significant degree. This test environment will likely be a simple javascript function that repeatedly queries the database for names using random input values, then runs those through the randomization algorithm. When this loop is done executing, we use an if-else statement to determine if any name exceeded the tolerance threshold for frequency. Specific UAT:
    - A user who enters that their pet is female should not receive a male-gendered name
    - A user who skips all questions should receive a name pulled from the entire database of names
    - A user who skips a question about gender should receive a name pulled from EITHER the female name pool OR the male name pool
    - A user who enters the same results on the quiz should not receive the same name more than a certain tolerance threshold (see definition above)

Individual Contributions:

- Owen Smith:
    - Writing all server-side scripts for the quiz system. Designing system for acquiring and storing user input and querying from name database. Currently working on script for randomization of final name list in JS.
    - GitHub Commit: https://github.com/CSCI-3308-CU-Boulder/3308Summer21_300_7/commit/ab8524325ca027511c8da4bdb3a17d143a1f4c56
- Jingqi Liu(JanesyLiu):
    - Tested and checked the questions files and projects files.
    - Will do web testing this week.
- Mathew Kim:
    - Currently working on the contact page, still have to make a dummy account to connect the contact page to.
    - Will finish up the email page this week

- GitHub Commit: https://github.com/CSCI-3308-CU-Boulder/3308Summer21_300_7/commit/ca008dc1946353f84bf28c48cd8e7c6adaa62f75

- Patrick Taylor:
    - Created Heroku project database to store project files
    - Worked on back end design for name randomization
    - Created JS class to store user data
- John Hodgen:
    - Worked on back end design and implementation for name randomization and
    - Designed and created SQL to construct and populate our database
    - Created backend to be added to Heroku database