COSC 310 Milestone 4

Anthony Argatoff, Aayush Chaudhary, Varun Patel, Dylan Pickard, Ryan Pybus

Testing implementation so far:

1. Writing unit tests for all python code using the unittest and the PyTest framework. This includes all classes and methods that we have created. Testing is non-trivial and covers all branches of code.
2. Visually inspecting web page elements to ensure elements appear in their correct positions. We test different viewport sizes to ensure that screen size changes do not break the application.
3. We will be improving testing in the upcoming weeks. We have reorganized our project file structure and have begun github continuous integration testing in our current sprint cycle.

Functionality that has been tested so far:

1. All of the functionality that has testing that is passing. This list includes:
    - Receiving data from our API
    - Populating the database with the data from the API
    - Testing persistence and set-up of the database
    - Database functionality (adding users, deleting users, modifying users, validating users, getting user info, validating admin accounts)
    - Creating and storing user notification settings as strings in the database
    - A control method to determine if a notification should be sent to a user, given the new event data from API
    - Calculating the distance between 2 geographical coordinate points
    - Sending emails containing a body and subject to a given address
    - Calculating a prediction for future earthquakes
    - Data is being displayed on our map page from the database
2. Front-end visual testing
    - We have been viewing each page as it has been added to the site, and checking links properly navigate between pages
3. Coverage on flask app is 88%, coverage on modular components is 86%

Automation so far:

1. We have implemented project automation. When new issues are added, they are automatically added to the sprint backlog and placed in the “backlog” column. When a pull request is made from an issue, the issue is automatically placed into the “In review”

column, and when the pull request is approved, the issue is closed and is moved to the “Done” column. This automation is working very well and saves a lot of time because we don’t have to manage each section of the kanban board. It is a major quality of life improvement.

2. We have added continuous integration pipelines with GitHub actions for setting up test automation. However, we are currently facing an issue with one of our tests, where the restrictions placed on the time range by the operating system are causing it to fail. (only the one test is failing, and doing so consistently, while it passes when done in isolation) Currently anytime a team member pushes their code into github, github will automatically run all tests and create a report of tests run.

Project summary so far

Our process

We have been using a mixture of agile software engineering development processes, including scrum and extreme programming. While we cannot truly perform either (since we cannot have daily scrum meetings, and we do not have any contact with the product owners or customers), we do follow the cycle aspects. Each sprint cycle (a week), we assess our backlog and decide what are the most important issues to bring to the next sprint cycle. Upon completing a sprint cycle, we evaluate our progress, determine what the next main issues would be, and dive into the next sprint cycle. As of right now, our process is reliable and clear. Nothing needs to change.

Ensuring code quality

We ensure code quality with 2 methods. Our first method involves feedback on pull requests. After a team member has developed a new feature, they will submit a pull request. At least 2 other team members review the code changes and run the code on their own machines, ensuring the tests still pass and the code still performs as expected. This is a great way to improve our code quality. We can catch small bugs or inconsistencies in our code before we commit it to our main branch.

Our second method is by giving demos in our team meetings. Each team member will give a quick demonstration of the feature they’ve created. They show how the new feature can be used, and what purpose it has in the project. This ensures that team members do their preparation and have something they can show the other team members for the meeting. This can improve our quality, as the group member will receive meaningful feedback and can make the appropriate changes.