Software Testing Assignment

BLACK BOX

Author: Aidan Pare

Test Case #1: Clicking the options on the left sidenav bar to change data sets. After an option is clicked, hover above any county or state, and the top right info card should state which data set you're in.

For example, click on State Cases, hover above California, and the top right info card should be titled "State Cases".

CONDITION	YES	NO
State Cases	X	
States Deaths	X	
Counties Cases	X	
Counties Deaths	X	

Summary: This was able to be run with no issues. The results were what we were expecting. The difficulty was easy, as it didn't take long to click on our four options and verify the data set. This was also done with multiple refreshes to make sure it loaded properly each time.

Author: Frances Watson

Test Case #2: Checking to make sure our total US COVID-19 cases and deaths numbers on the site's header matches with NY Times data.

This is done by looking at total cases and deaths for the displayed date on the NY Times GitHub file (our source for our database). This was a three-day test, since we wanted to also test our auto-update feature which happens daily.

CONDITION	YES	NO
2020-12-10	X	
2020-12-11		X
2020-12-12	X	

Summary: This was an easy one, we just wanted to check for accuracy. Our auto-update feature updates every day at 2:30AM EST. The second day had an error because the update time of our auto-update feature wasn't aligned properly with our source's. This forced us to update the time for our auto-update feature to be more flexible, and we ended up making it check for the date from two days ago, instead of just one. This allowed us to correctly display it for our third day of testing.

Author: Jennifer Nguyen

Test Case #3: Verifying the legend colors match the numbers shown when hovering above a state or county.

This was done for each state, and some counties due to the amount of them.

1ST ITERATION

CONDITION	YES	NO
State Cases		X
States Deaths		X
Counties Cases		X
Counties Deaths		X

2nd INTERATION

CONDITION	YES	NO
State Cases	X	
States Deaths	X	
Counties Cases	X	
Counties Deaths	X	

Summary: This test became time consuming and a bit difficult due to the number of counties and how small they are. Moving around the map along with zooming in and out isn't the smoothest either. Therefore, we decided it was best just to do all of the states, and only some of the counties. The counties chosen were 20 from each color in the legend (or less if that many didn't exist). The All four conditions failed initially failed. This was found because bigger states that were known to have more cases and deaths had lighter colors than smaller states with smaller numbers. After looking into it, it seemed that the issue was with a built-in function within the choropleth class within Leaflet, as it was scaling the colors wrong with the data. In the end, we decided to scrap using the built-in function and we made our own function using own custom colors with hard coded ranges. We tested all four conditions with a 2nd iteration with this new function, and all of the conditions' colors showed correctly.

WHITE BOX

Author: Karim Durrani

Test Case #1: Checking to make sure all of our REST API scripts were operating correctly. These scripts were reading from our GitHub sources, formatting the data, and storing into our database. The other half is reading from our database, parsing through our geojson files and updating the information within them with the data from our database.

1st ITERATION

CONDITION	YES	NO
rest_api_us.js	X	
rest_api_counties.js	X	
rest_api_states.js	X	
rest_api_prisons.js		X

2nd ITERATION

CONDITION	YES	NO
rest_api_us.js	X	
rest_api_counties.js	X	
rest_api_states.js	X	
rest_api_prisons.js	X	

Summary: This test took quite a bit of time because of the issues we were running into. All of our data sources were in a csv format, which made it easy to parse and store into the database. The only one giving us issues was our prison data set. This got a NO for our 1st iteration because one of the columns inside the csv was having issues with how we were parsing. The last column called *notes* had commas and newlines, which is how we were parsing the csv file. Since the notes weren't important to us, we were able to figure out a way to ignore that entire column and continue parsing.

Software Testing Assignment

Another error we were getting mentioned incorrect values not matching the column values. After some research, this was apparently caused by the initial type of the declared variable inside the MySQL table during its creation. We ended up changing the majority of the variable types to VARCHAR, which fixed the entire issue and enabled them to all work properly during our 2nd iteration.

Author: Bobby King

Test Case #2: Checking to make sure all of geojson updating scripts were operating correctly. These scripts were reading from our database, parsing through our geojson files and updating the information within them with the data from our database.

1st ITERATION

CONDITION	YES	NO
runPythonUS.js	X	
runPythonCounties.js		X
runPythonStates.js		X
runPythonPrisons.js		X

2nd ITERATION

CONDITION	YES	NO
runPythonUS.js	X	
runPythonCounties.js	X	
runPythonStates.js	X	
runPythonPrisons.js		X

Summary: This was another test that took quite a bit of time because of the issues we were running into. Originally, we opened the geojson files inside Visual Studio Code to do a replace all for certain features to rename them deaths and cases, to make it easier to test and call data. When we tried to use these files at the initial files for data, we kept getting errors. We noticed that it worked fine if we didn't open the file and do a replace all for terms, so we thought it wasn't replacing them all. Apparently, the actual issue was that if it was opened in VS Code, VS Code had issues display all characters, so when it was saved, it didn't save everything. Instead, we did a replace all inside a text editor, saved, and it ended up working.

Software Testing Assignment

The majority of the scripts worked during our 2nd iteration, but the prison data set wasn't populating enough data for some reason. After a lot of time with testing, we determined that the data set wasn't good with how we were querying data. Essentially, we would calculate the proper date, then put that into part of the query to grab all data from that particular date. The prison data didn't have this information always updated, and only listed updates when it changed. For example, if we tried to query for all deaths from a certain date, instead of giving us the deaths from each prison for that date, it would only give us deaths for prisons that changed that date. This meant we would have instances where we only got two results instead of every prison. The solution to this would be to create a table, not drop it each time we update information, and create a function to update the table but keep all pre-existing data, to match our system. Unfortunately, this was discovered too late and the prison data had to be scrapped from the project.