Module 6 Assessment

Introduction

It's time to practice and solidify the skills you learned in this module.

Warning: Independent work

This is an individual assignment. Please do not collaborate with your peers or share your work until the project is reviewed as a class.

Setup

1. Click here to download the code you need to for this project.
2. Navigate to the folder you just downloaded on the command line.
3. Get dependencies installed by running `npm install`.

Part 1: Server Setup

Start with setting up your server to serve static files so that it will be ready for deployment.

- Near the top of `server.js`, set up middleware and/or endpoints that will serve the files from the `public` folder.
- Don't forget the CSS and JS files, you can see the endpoints they're requesting in `public/index.html`.
- Start the app by running `nodemon`.
- **Visit `http://localhost:3000` your web browser and play a few games of Duel Duo.**
  Familiarize yourself with how the interface works. The idea is that the player's Duo and the computer's Duo will duel each other. You don't know the computer's Duo beforehand. The winner is calculated by adding each Duo's total health and total attack damage up and then subtracting all of the opposition's attack from the defender's health. (Reloading the page will reset your wins and losses.)
- **Open and read *public/index.js*. While it's not important to**
  understand every line of how the JavaScript code is working in this project, take note of the different functions and what they appear to do to make the app work. Also, explore more of your server so you know what's going on there too.
- Remember, you can kill the nodemon process by pressing `ctrl + c`

Part 2: Manual Testing

There are at least 3 known bugs in this game. Draft a plan to test the game, conduct your testing, and document any bugs that you find. Keep track of your testing documentation (plan, cases, and bugs) in a Google doc, Trello, or something similar.

Make sure you play enough times to lose at least once and win at least once.

You need to write at least 1 test case, 1 bug report, and 1 other piece of documentation (a test plan, another case, or another bug report).

When you have finished this section, download your document as a PDF/png, and save it to your project folder.

Part 3: Unit Tests

In the `functions.test.ts` file, there is a describe block waiting for tests!

Write **at least 2** tests in the describe callback for the `shuffleArray` function. You may need to read through Jest documentation to find methods/matchers to use.

Some ideas:

- check that `shuffleArray` returns an array
- check that it returns an array of the same length as the argument sent in
- check that all the same items are in the array
- check that the items have been shuffled around

Once you're done, make sure your server is running with `nodemon`.

Next, in a **new terminal window** while your server is still running, run the tests with `npm run test`. You should see ***1 passed*** in green when this command runs.

Part 4: Write Automated Tests

Open and read the file called ***duelDuo.test.ts***. There is one automated test included already.

Write at least 2 more tests for the game. Here are some ideas for tests:

- Check that clicking the Draw button displays the div with id = "choices"
- Check that clicking an "Add to Duo" button displays the div with id = "player-id"
- Check that when a bot is "Removed from Duo", that it goes back to "choices"

Note: You may want to use the `sleep` function after clicking on an element to make sure the tests don't get ahead of themselves.

You can run these with `npm run test` as well.

Part 5: Rollbar

Set up Rollbar in `server.js` so that you can log information and errors about your app.

In the Rollbar app, create a new project to get your access token. Use that to connect your app to that project.

In the endpoint handler functions, set up at least 4 rollbar events (not counting 'Hello World'.)

Part 6: Deployment

- Set up your Procfile.
- Add and commit your code.
- Set up a repo in GitHub.
- Connect your remote repo and push your code.
- In Heroku, set up a new app that deploys from your repo.
- Include the link to your Heroku site in the `heroku.md` file.
- Interact with your live site and check Rollbar for logs!

Extra Credit: Deployment Sketch

Draw a sketch/picture of deployment as it relates to the broader web application development process. Be sure to include GitHub, your local development environment, and the different parts of the web application on the sketch.

You should be able to come back to this picture in the future if you need a refresher on what deployment is and how to think about it in the broader picture of web application development.

Be sure to push your code to GitHub for this assessment!

To pass this assessment you must score at least 17/24.

OUTCOME	DETAIL	0 pts	1 pt	2 pts	3 pts
Server Setup and Deployment	The server is set up to serve static files, an accurate Procfile is included, and a working link to the hosted site is given.	Did not attempt.	Attempted but none of the requirements are complete and/or they are incorrect.	1-2 requirements and complete and correct.	All 3 requirements are complete and correct
Manual Testing	Assessment includes at least 3 pieces of documentation	Did not attempt.	Assessment contains 1 piece of documentation	Assessment contains 2 pieces of documentation	Assessment contains 3+ pieces of documentation
	Assessment includes one test case. The test case contains 1) a description of the test, 2) the steps of the test, and 3) the postconditions.	Did not attempt.	Attempted but none of the sections are complete.	1-2 of the sections are complete and correct.	All 3 sections are complete and correct.
	Assessment includes one bug report. The bug report contains at least 1) a description, 2) steps to reproduce, 3) expected result, 4) actual result, and 5) environment sections.	Did not attempt.	Bug report exists and contains 1 - 2 of the required sections	Bug report exists and contains 3 - 4 of the required sections	Bug report exists and contains 5+ of the required sections
	Assessment contains a 3rd piece of documentation. A test case or bug report follows the criteria above. If it is a test plan, the test plan includes 1) an overview, 2) test criteria, 3) entry criteria, 4) exit criteria, and 5) details sections.	Did not attempt.	Test plan includes 1 - 2 of the required sections	Test plan includes 3 - 4 of the required sections	Test plan includes 5+ of the required section
Unit Tests	Assessment includes at least 2 tests for the shuffleArray function.	Did not attempt.	Attempted but incomplete and/or incorrect	Assessment includes 1 complete and correct test	Assessment includes 2+ complete and corre tests
Automated Tests	Assessment includes at least 2 additional syntactically correct automated tests	Did not attempt.	Attempted but the tests are incomplete and/or incorrect	Assessment includes 1 additional syntactically correct automated test	Assessment includes 2+ additional syntactically correct automated tests
Rollbar	Assessment includes Rollbar and at least 4 Rollbar events	Did not attempt	Attempted but Rollbar setup is incorrect or there are fewer than 2 events	Assessment includes correct Rollbar setup and 2-3 events	Assessment includes correct Rollbar setup and 4+ events
Extra Credit	Assessment includes a deployment sketch that accurately depicts deployment and includes the following: GitHub, your local dev environment, and the parts of the web app	Did not attempt.	Sketch is inaccurate and/or incomplete	Sketch is accurate but incomplete	Sketch is accurate and complete
Total		**x/24**			

© 2021 Devmountain