Capstone Code Review Checklist

This document is a code review checklist for our capstone project. We followed the effective code review checklist example suggested by Tim Maciag that can be found here: https://nyu-cds.github.io/effective-codereviews/. The purpose of this checklist is to comment on the current state of our code both the good and the bad as well as to try and eliminate any mistakes or bad practices if we can.

General

1) Does the code work, perform its intended function, and have correct logic?

    Yes, all of our code compiles, works, and does its intended functions. This is one of the strengths of our application as we traded off aesthetics and flashy features for code that performs all of its functions well and correctly. We performed unit testing on our functions and have implemented error checking and string escaping to prevent incorrect or malicious data being inputted into our system.

2) Is all of the code easily understood?

    We feel that our code is generally easily understood. We used a consistent spacing and indexing scheme when writing our application, and provided detailed comments that explain our more complex functions and data structures. All related HTML elements are grouped together and have a consistent blocking scheme that makes it easy to follow.

3) Does it conform to your agreed coding conventions?

    As mentioned in the previous question, we kept our spacing and braces consistent throughout all pages. It helped both us and others when trying to understand and read our code. We had a variable naming convention of including the name of the page within the variable name if needed. This helped us make sure we were using the right CSS on similar elements because we encountered conflicting CSS during the testing process. We commented all complex functions, data structures, and HTML elements as well as kept all code

lines under one screen width so that minimal horizontal scrolling is needed when reading the code.

4) Is there any redundant or duplicate code?

We do not have any complete duplicate code, but we do have elements in our forum pages that are very similar. Some of the CSS spacing is repeated, but new CSS classes were still necessary due to CSS conflicts that arose when we tried to re-use the CSS we had already written. We had conflicts that caused the CSS of different pages to load differently based on which pages were loaded first. For this reason we chose to re-use the spacing, but create new CSS classes to avoid this issue.

5) Is the code as modular as possible?

Vue.js is modular by nature because you can create small component modules that can be reused throughout your application, and since we used vue-router we created a collection of smaller individual pages that make up the entire application as a whole. In terms of our functions and methods, we kept them all small and performing one task only except for one. We have a function called getInsurers() that could be potentially broken up into smaller modules. We made the choice to keep this function larger because we were running into timing issues during the testing phase. Originally we had separate functions being executed, but the results of one function were required for the execution of the next. Since these functions used HTTP requests to our database, our application was trying to execute the next function before the response from our database was complete resulting in errors. By moving everything into one function, it eliminated the timing issues, but created a less modular function.

6) Can any global variables be replaced?

We used very little global variables in our application. The only global variable that we have is the navigation bar which is required/displayed on every page of our application so it cannot be replaced.

7) Is there any commented out code?

No, there is no commented out code remaining within our application. There was commented out code during the development stages of our application, but has been removed for the final product.

8) Do loops have a set length and correct termination conditions?

Yes, all of our loops have been unit tested with different amounts of data being pulled from our database and all tests succeeded. All of our functions that rely on data responses from our database are dynamic and we have no issues in terms of terminating loops within our application.

9) Do the names used in the program convey intent?

Yes, we used descriptive variable names throughout our entire application. If the variable name was for an HTML element, we used names that described its purpose as well as including the name of the page it was being used on within the variable name. For javascript elements we used variable names that accurately described the data being stored inside that particular variable. The names used for methods and functions were ones that described the action trying to be accomplished by that particular function.

Performance

1) Are there any obvious optimizations that will improve performance?

The only optimization that could be implemented into our current application is a loop to calculate the quotes from our table. Currently it is done by indexing because we have a small sample size of insurers to get data from. The actual speed of the program would most likely not be affected very much by this change, but it would clean up the code a bit and make it easier to read. We didn't implement this change due to time constraints and not wanting to change our working product in case something ended up not working after the change.

2) Can any of the code be replaced with library or built-in functions?

    Since most of our functions are custom functions that contact our database, nothing obvious can be replaced with built-in functions or libraries.

3) Can any logging or debugging code be removed?

    All logging and debugging code has already been removed from our final source code.

Security

1) Are all data inputs checked and encoded?

    All of our input boxes are set to the proper data types and checked for errors before they are used in our functions. Also, when the MySQL statements are executed in our Express.js API, all inputs are escaped using the mysql.escape() function to protect against SQL injection and cross site scripting. However, none of our data is encoded and is one of the main drawbacks of our application. Our functionality is very strong, but the security of our application could use improvement.

2) When third-party utilities are used, are returning errors being caught?

    When accessing our API, all data is checked before being inputted which helps avoid errors and all responses from the API are returning known values from our database. However, if in invalid input is somehow inputted into the database, if it is an invalid type an error is thrown and is not accepted into the database.

3) Are output values checked and encoded?

    All of our outputs are outputted into the browser so checking the values is somewhat redundant since it is visible. We tested our output values until we were getting correct outputs.

4) Are invalid parameter values handled?

    Yes, we have checked for, handled, and provided error messaging back to the users of our application when invalid inputs are present. Handling the inputs before they are even accepted allows us to prevent a lot of unnecessary queries to the database and also prevents invalid data from entering the database.

Documentation

1) Do comments exist and describe the intent of the code?

    Yes, we do have comments that describe parts of the HTML structure as well as the intent of our functions, methods, and data structures.

2) Are all functions commented?

    Yes, all functions have been commented.

3) Is any unusual behaviour or edge-case handling described?

    Our more complex functions that are not immediately understood are commented more heavily to provide clarification on what the function is trying to achieve or what a data structure is for. We don't have very many edge-case handling except for some very simple date checking.

4) Is the use and function of third-party libraries documented?

    The only third party library that we use is axios. Not much documentation was needed because all it does is allow GET and POST requests to our Express API.

5) Are data structures and units of measurements explained?

    Our complex data structures and dynamic arrays that store data retrieved from our database are explained within the code. All of the arrays are commented explaining what kind of data is expected to be held in them.

6) Is there any incomplete code?

No, we do not have any incomplete code and our application has full functionality based on what we wanted to accomplish with this capstone. More could be done to improve our application in the future, but is out of our scope for this project.

Testing

1) Is the code testable?

All of our testing was done using the Vue developer tools chrome extension and console logging. Vue developer tools allow you to view the HTML structure of the current view being displayed as well as all data structures and their values for that current page. You can also console log any information that you would like to see and view it using the Vue developer tools extension. We were able to test all responses from our database and the values of parameters after functions were executed which served our purposes.

2) Do tests exist, and are they comprehensive?

As mentioned above, there are no coded test cases, but we did comprehensive testing using Vue developer tools to confirm our responses from the database and the values that users were inputting.

3) Do unit tests actually test that the code is performing the intended functionality?

As mentioned above, we don't have formal coded unit test cases, but we were able to confirm the results being used by our application matched the data within our database. Our functions were also calculating correct results for user quotes tested on multiple different insurers with different prices.

4) Could any test code be replaced with the use of an existing API?

    We currently use our Express API to retrieve all data and we were able to test the data retrieved by viewing it in the browser and confirming that it was the same both in the browser and our database.