EzParking Myadmin

Code Quality Report

SSE Capstone project – University of Regina

Team Apollo

Winter 2023

Yilin Ren

Executive Summary

This code quality report provides an assessment of the EzParking MyAdmin web application, including an evaluation of its codebase, testing methodology, security, documentation, and overall code quality, as well as suggestions for improvement.

The EzParking MyAdmin application is built using the widely used React.js framework, which provides a powerful and flexible platform for building high-performance user interfaces.

This report provides an in-depth analysis of the EzParking MyAdmin app, examining whether the application was built with quality in mind, following coding standards and best practices, and avoiding common pitfalls like security holes and code complexity.

Table of Contents

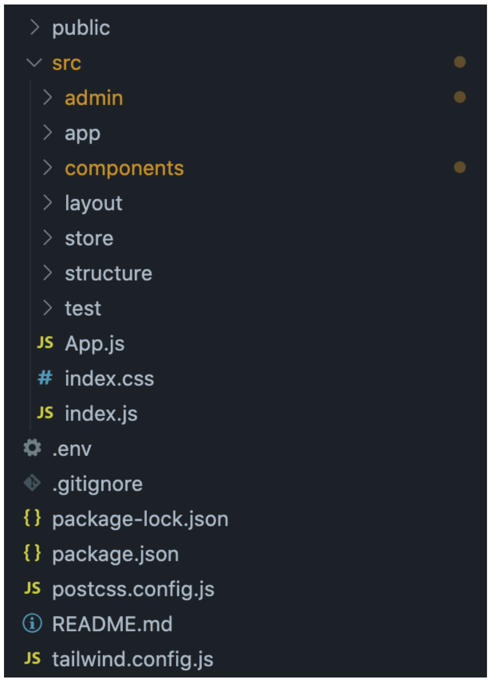
Code formatting

Maintaining consistent indentation is crucial to ensure code readability. In certain languages, such as Python, indentation serves as a critical means to distinguish code blocks. By analyzing MyAdmin's code, we can observe its consistent use of two tabs for indentation, which significantly enhances its readability.

Simultaneously, all variables use the camel case naming convention, and some highly repetitive string variables are also normalized in the way of “export default const”. This ensures that implicit type conversions do not occur when comparing strings and eliminates the chance of bugs due to misspellings.

Code Architecture

The source code of this project is organized according to the function to which the code belongs and stored in different folders. The Admin folder saves code related to login, logout, and user sessions. The App and Store folders hold code related to the runtime state of the program. The Components folder holds components that can be reused on multiple pages, such as navigation bars. The Structure folder holds the class files for storing the data structure of the path graph. The Test folder holds all test scripts. This helps developers or future users easily identify the directory of the file they are looking for.

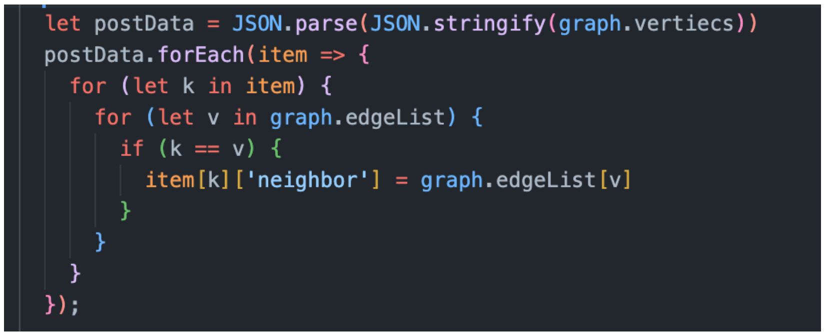


```
> public
v src
  > admin
  > app
  > components
  > layout
  > store
  > structure
  > test
  JS App.js
  # index.css
  JS index.js
.env
.gitignore
{} package-lock.json
{} package.json
JS postcss.config.js
README.md
JS tailwind.config.js
```

Figure 1. Program file structure

Coding Practice

When the program needs to use some variables, such as when the request sent contains a binary variable, this variable should not be hardcoded because it will cause serious unreadable consequences. The correct approach is to use a suitable serializer to generate this binary variable. There is a similar usage in the EzParking MyAdmin program, which uses the JSON library function to parse the JSON data structure passed from the backend instead of splitting the string directly.



```
let postData = JSON.parse(JSON.stringify(graph.vertiecs))
postData.forEach(item => {
  for (let k in item) {
    for (let v in graph.edgeList) {
      if (k == v) {
        item[k]['neighbor'] = graph.edgeList[v]
      }
    }
  }
});
```

Figure 2. Parse JSON data to object

Non-Functional Requirements

JSX, which can be described as the HTML language embedded in React components, is an interpretive language used by web development. It has excellent readability and follows the Open-Closed theory. When a JSX component is compiled by webpack, webpack will verify and guarantee that each tag is closed, ensuring later readability and maintainability in the long term. In the EzParking MyAdmin application, the source code was interpreted by Babel and webpack, preventing the application from having any syntax errors.

Testing

React.js offers another advantageous feature: each UI is treated as a component. It is not difficult to find that in the MyAdmin program, many repeated components have the same style. These components apply the same template, only the variables in them have been changed. Then, because these components use the same template, they are remarkably easy to test. It only requires a single test code to evaluate multiple components.

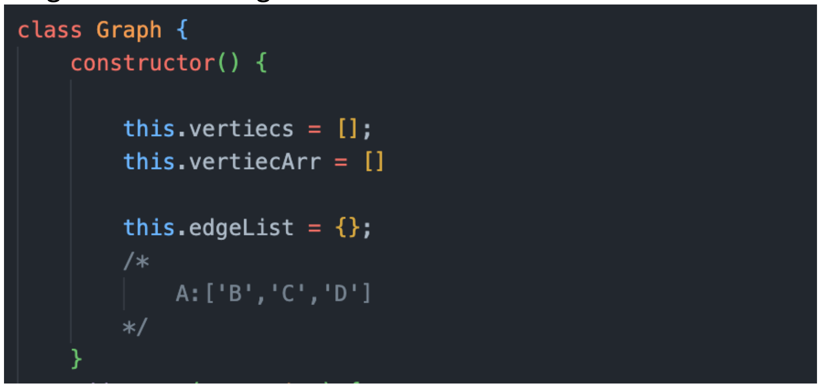
Performance

The MyAdmin program strictly abides by the DRY (Do not repeat yourself) principle. If there is any chance that a component needs the same or similar code to complete, it is written as a new independent component in the MyAdmin program. Components such as TopSection.js and InstructionSection.js have been split from the main project as an example.

If a page is always stuck at runtime, it will result in a poor user experience. When writing the MyAdmin program, the performance of the entire page was always kept in mind. In the MyAdmin program, users will set many waypoints. These waypoint objects generally save the corresponding coordinates and attributes. If they are simply saved in the form of a blob or array, there will be a noticeable lag when the user deletes the waypoint. This is due to the underlying logic of the browser's JS engine and cannot be avoided.

In Chrome's V8 engine, if you want to delete the waypoint in the middle of an array, you need to move the index of each waypoint forward in the array. This would be very browser resource intensive.

The method used by the MyAdmin program utilizes the newly introduced ES6 technique, class, and constructor. Waypoints are added to the graph class as objects. The waypoint object is stored in a form similar to a linked list in the graph class, thus avoiding the inherent disadvantages of the V8 engine.



```
class Graph {
    constructor() {

        this.vertiecs = [];
        this.vertiecArr = []

        this.edgeList = {};
        /*
            A:['B','C','D']
        */
    }
```

Figure 3. Usage of ES6 feature of Class and contructor

Conclusion

This code quality report provides a comprehensive analysis of the Ezparking MyAdmin application. The report covers the assessment of the codebase, program architecture, programming practices, testing, and performance. During the analysis, we found some potential problems such as code readability issues, potential performance issues, and so on. At the same time, we also found some advantages, such as a clear code structure.

To improve the quality of the MyAdmin application, we provided some advice, such as optimizing documentation and increasing test coverage. By implementing these improvements, developers can enhance the overall quality of the application and contribute to a more scalable application.