Code Quality Report

Repository: Freight Shield

Performed By: Codacy

Coding Standards and Conventions

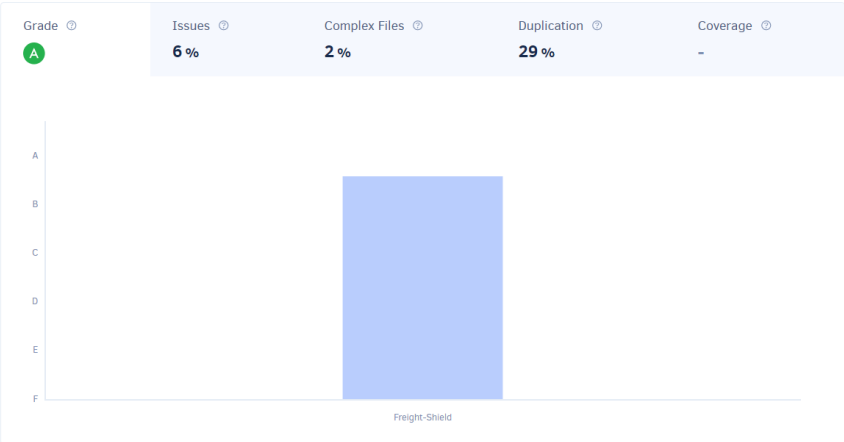
Our project adheres to the principle of "Separation of Concerns," structuring the filesystem into distinct sections for the frontend, backend, and mobile development. This organizational strategy is extended to accommodate additional directories that manage Docker images, ensuring a streamlined workflow.

We employ camelCase naming conventions across our files and directories to maintain consistency and avoid issues related to case sensitivity that may cause untracked file errors in Git. Specifically, within the backend, we further refine our structure by categorically organizing code into dedicated folders for middleware, controllers, models, and the server configuration.

This systematic approach to our filesystem and naming conventions not only enhances readability and maintainability but also aligns with best practices that facilitate integration and deployment processes on platforms like GitHub.

Code Maintainability

Within each controller module, we meticulously document every method, specifying the route it pertains to and the specific action it performs. This includes a detailed description of the method's role within the application, the logic it encapsulates, and its interactions with other components.



Codacy review of the repository

Grade: The Grade "A" conferred by Codacy represents the pinnacle of code quality within our project. This prestigious rating is determined by a composite of various metrics, which include the presence of issues, code complexity, instances of code duplication.

Duplication: Duplication are due to the fact that application structure is the same with different styling. This led to duplication of the code. In a way the code is still DRY but due to the code being very massive, it will detect duplication in files that contain forms, buttons etc.

Issues in the code

Security: Review has uncovered a significant security risk within the controller functions responsible for user authentication. The method by which users are identified—through cookies containing email addresses or user IDs which presents a vulnerability to NoSQL injection attacks. This weakness could potentially allow unauthorized individuals to forge cookies using known email addresses and passwords to gain access.

Code Style: This originated primarily from the presentation and formatting within the README file. While these do not directly impact the operational codebase, they highlight the need for consistency in our documentation. Additionally, the codebase exhibits instances of extra bracket use.

Error Prone: The analysis has detected error-prone patterns, specifically concerning unused variables in the codebase. These variables can lead to confusion and potential errors in the maintenance phase

Code Performance

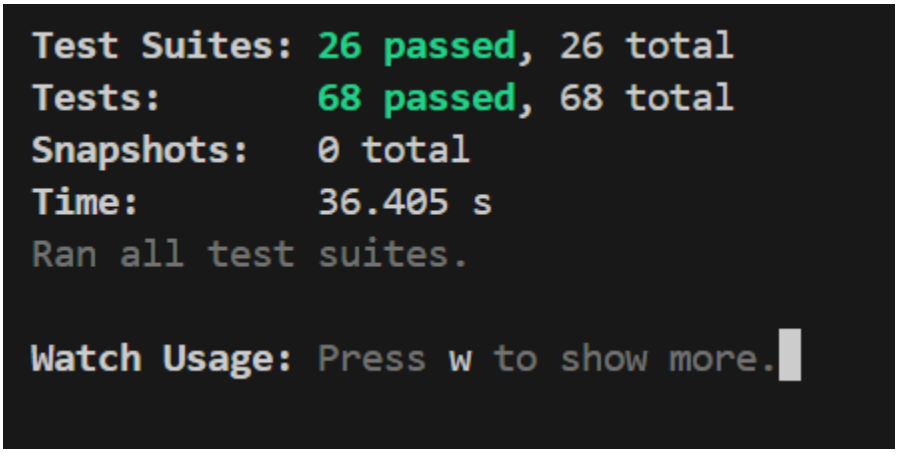
The performance metrics for our codebase are highly satisfactory, attributed significantly to our dedicated server infrastructure. The optimized design of our request handling, which ensures that only necessary data is encapsulated within objects and transmitted, contributes to this high-performance level.

However, a performance bottleneck has been identified related to the integration with external APIs, specifically the News and Google APIs. The extensive volume of data retrieved from these services poses efficiency challenges. The high latency in data fetching can impact the user experience and increase the load on our systems.

Testing

Unit testing has been rigorously conducted with the Jest framework, which is well-suited for testing React applications. The Jest library provides a robust platform for validating the integrity and correctness of new code, ensuring that components behave as expected. During these tests, we focused on component functionality, state management, and event handling to confirm that individual units perform correctly in isolation.

Complementing our automated unit tests, UI and Usability testing were carried out manually. This hands-on approach allowed us to critically assess the application from the end-user's perspective, ensuring that the user interface is intuitive and accessible. We tested various aspects, including navigation flow, element alignment, color schemes, and interactive responsiveness, to validate the overall user experience.



Unit Testing performed using JEST

Dependencies

The dependencies of the code base are up to date and constantly checked if something is being installed that is not required. This leads to clean code and function the dependencies appropriately.

Architecture and Design

The design follows MVC which was intended from the get go. With MVC, the succession of MERN stack is implemented.

Conclusion

The attainment of a Grade "A" in our code quality assessment is a testament to industry standards and best practices. This distinguished rating affirms the high caliber of our codebase, reflecting a disciplined approach to development that prioritizes clarity, maintainability, and security.