EzParking service

Code Quality Report

SSE Capstone project - University of Regina

Team Apollo

Wintern 2023

Ziwen Tan

200394440

# Table of content

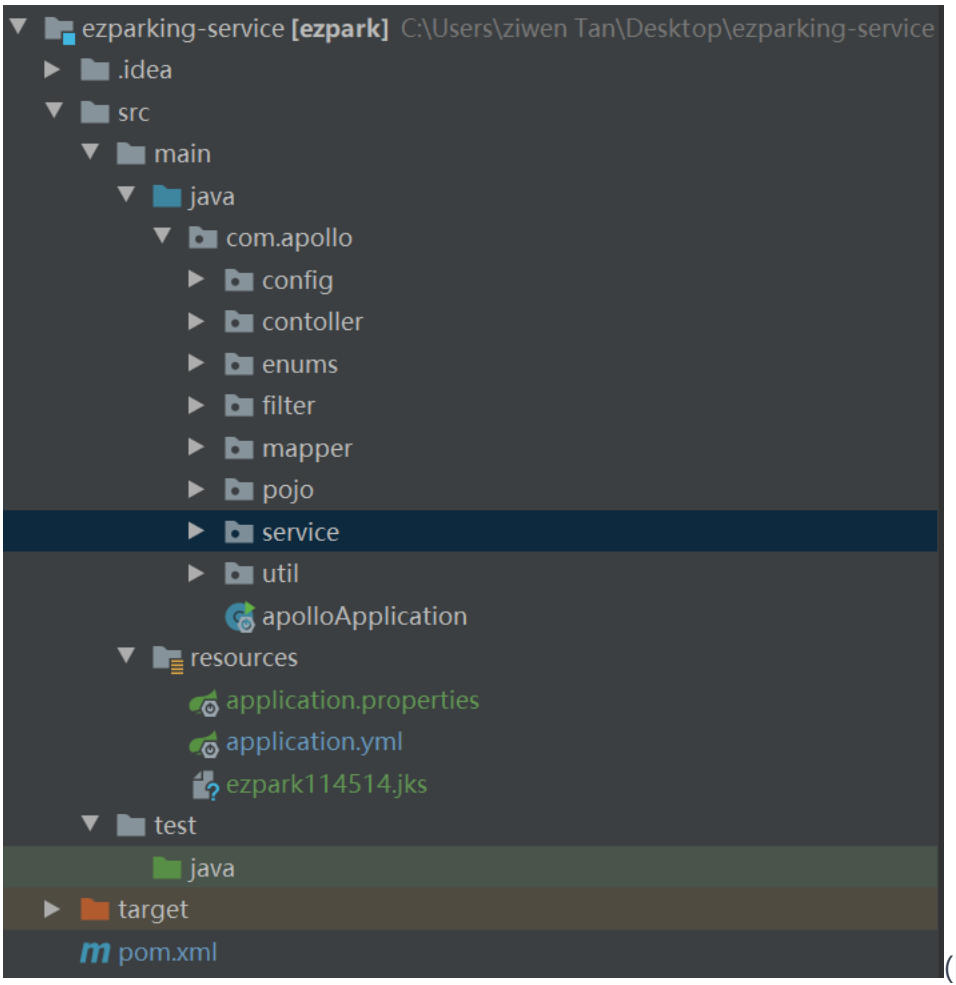
# Summary

The EzParking service provides data processing and storage functions for the entire Ezparking application, serving as a console, client, and AI detector for EZparking. This report focuses on evaluating the quality of the backend code. The evaluation will be conducted from four perspectives: code architecture, performance, security, and overall code quality. By examining these codes, we can comprehensively understand the quality of the backend code and identify areas that may be improved.

# Code Architecture

The backend code of the EzParking application is implemented using Java and a small portion of XML scripts for configuration, based on the Spring Boot framework. This choice of technology enables the implementation of the SSM (Spring, SpringMVC, MyBatis) architecture, which effectively separates the responsibilities of presentation, business logic, and data access. By adopting this architecture, the backend codebase can be easily managed and maintained, improving its robustness, scalability, and maintainability. In addition, the Spring Boot framework provides a wide range of tools

and features that simplify the development process, such as auto-configuration, an embedded web server, and easy integration with other libraries and frameworks. Furthermore, Maven is used for dependency management, allowing for the efficient management of external libraries and simplifying the project's build process. Overall, the combination of the SSM architecture, Spring Boot framework, and Maven dependency management results in a well-structured and easy-to-maintain codebase for the EzParking backend.



(Folder structure)

# Performance

During the development process, we used parallel streams, synchronized blocks, and various dependencies to ensure optimal performance. Parallel streams split transactions to achieve faster processing speed, while synchronized blocks ensure thread safety. In addition, we used HashMap to improve performance compared to traditional lists and linked lists. HashMap uses HashCode for queries, which is faster than searching through a list. Lists are fast for queries, but slow for modifying elements at the tail end. Linked lists perform well when there is little data, but their performance degrades with increasing data volume. HashMap combines the advantages of lists and linked lists, and when the number of elements in each index is greater than or equal to eight, it reconstructs the data structure into a red-black tree. By using HashMap, the shortest path search algorithm can save a significant amount of search time and improve performance. The Dijkstra algorithm can efficiently calculate the distance between each node.

# Security

The entire project's security is based on the Spring Security framework, which has many advantages, including its powerful authentication and authorization mechanisms and high degree of customizability. Spring Security provides rich authentication and authorization mechanisms, making it easy for developers to implement various security

scenarios, such as role or permission-based authorization, single sign-on, OAuth2, and more. Moreover, Spring Security is a highly customizable framework that can be tailored to the needs of the application, enabling more flexible and secure authentication and authorization mechanisms. Therefore, Spring Security is one of the preferred security frameworks for many Spring Boot projects. By using Spring Security, access to the control system and certain client APIs can be restricted, preventing data modification by unauthorized users and malicious access to project interfaces.

# Overall Code Quality

Developers will upload their code to non-main branches on GitHub, and when submitting new code to the main branch, they must create a pull request that is assigned to all other team members for review and approval. Through this process, every code upload must pass through the review of team members, greatly improving the quality of the code.

# Conclusion

Overall, the EzParking backend code has a robust architecture, optimal performance, strong security measures, and high-quality code. However, there is always room for improvement, and the team should continue to review and refine the code to ensure the best possible performance, security, and maintainability.