1. Introduction

The Rule Engine is designed to provide a flexible and generic mechanism for defining and applying rules to different types of objects. This document outlines the architectural requirements for the Rule Engine system.

2. Functional Requirements

2.1 Rule Definition

- **FR 2.1.1:** The Rule Engine shall support rule definitions using a JSON format.
- **FR 2.1.2:** Rules shall be dynamically created based on the JSON rule definitions.

2.2 Rule Conditions

- **FR 2.2.1:** Rules shall support conditions based on properties of the target object.
- **FR 2.2.2:** Supported conditions include checks for null, not null, and not empty strings.

2.3 Rule Actions

- **FR 2.3.1:** Rules shall support actions to be performed if conditions are met.
- **FR 2.3.2:** Supported actions include setting property values on the target object.

2.4 Rule Application

- **FR 2.4.1:** The Rule Engine shall apply rules to a target object.
- **FR 2.4.2:** The Rule Engine shall evaluate conditions and execute actions accordingly.

2.5 Generic Rule Engine

- **FR 2.5.1:** The Rule Engine shall be generic and applicable to different types of target objects.

2.6 Rule Parser

- **FR 2.6.1:** The Rule Engine shall include a rule parser (`JsonRuleParser`) to parse rule definitions from JSON.
- **FR 2.6.2:** The rule parser shall be extensible to accommodate different implementations for parsing rules from various sources.

3. Non-functional Requirements

3.1 Separation of Concerns

- **NFR 3.1.1:** The Rule Engine shall follow the principles of encapsulation and separation of concerns.

3.2 Error Handling

- **NFR 3.2.1:** The Rule Engine shall provide error handling for cases where conditions or actions cannot be invoked.

3.3 Flexibility

- **NFR 3.3.1:** The Rule Engine shall be designed for flexibility, allowing for easy extension or modification of the rule processing logic.

3.4 Dependency Injection (DI)

- **NFR 3.4.1:** While not explicitly implemented, the Rule Engine shall be designed to support potential Dependency Injection (DI) patterns in the future.

4. Conclusion

The Rule Engine architecture shall fulfill the outlined functional and non-functional requirements, providing a robust and flexible solution for rule definition and application.