1. Rule Definition Parser:

- **Responsibilities:**
  - Parse JSON rule definitions.
  - Translate JSON rules into internal rule objects.
- **Implementation:**
  - Utilize JSON.NET library for JSON parsing.
  - Define a `RuleDefinitionParser` class with methods to parse rules.
- **Interfaces:**
  - `IRuleDefinitionParser` interface with methods like `ParseRuleDefinitions`.

2. Rule Engine Core:

- **Responsibilities:**
  - Manage a collection of rules.
  - Evaluate and apply rules to target objects.
- **Implementation:**
  - Implement a `RuleEngineCore` class.
  - Maintain an internal collection of rules.
- **Interfaces:**
  - `IRuleEngineCore` interface with methods like `AddRule`, `EvaluateRules`, and `ApplyRules`.

3. Rule Executor:

- **Responsibilities:**
  - Execute actions associated with rules.
  - Apply changes to target objects.
- **Implementation:**
  - Implement a `RuleExecutor` class.
  - Define methods to execute actions based on rule conditions.
- **Interfaces:**
  - `IRuleExecutor` interface with methods like `ExecuteAction`.

4. Target Object Interface:

- **Responsibilities:**

    - Define properties and methods required by rules.
- **Implementation:**
    - Application-specific classes implement this interface.
- **Interfaces:**
    - `ITargetObject` interface with properties and methods required by rules.

5. Rule Engine API:

- **Responsibilities:**
    - Expose APIs for adding rules, evaluating rules, and applying rules.
- **Implementation:**
    - Define a `RuleEngineAPI` class.
    - Implement methods to interact with the Rule Engine Core.
- **Interfaces:**
    - `IRuleEngineAPI` interface with methods like `AddRule`, `EvaluateRules`, and `ApplyRules`.

6. Dependency Injection (DI):

- **Implementation:**
    - Use a DI container to manage dependencies between components.
    - Inject dependencies (e.g., `IRuleEngineCore`, `IRuleExecutor`) into the Rule Engine API and other components.

7. Security Module:

- **Responsibilities:**
    - Implement access controls for rule definition inputs and API methods.
    - Enforce secure coding practices.
- **Implementation:**
    - Utilize .NET security features.
    - Implement role-based access controls.

8. Logging Module:

- **Responsibilities:**
    - Capture rule evaluation and execution events.
    - Generate logs for monitoring and debugging.
- **Implementation:**
    - Implement a `Logger` class.

    - Integrate logging statements within rule engine components.

9. Testing Module:

- **Responsibilities:**
    - Implement unit tests for individual components.
    - Develop integration tests for rule evaluation and execution.
- **Implementation:**
    - Use testing frameworks like NUnit or xUnit.
    - Develop test suites for different scenarios.

10. Performance Optimization:

- **Responsibilities:**
    - Optimize rule evaluation algorithms for performance.
    - Cache frequently used rules.
- **Implementation:**
    - Utilize efficient algorithms for rule evaluation.
    - Implement a caching mechanism for rules.

11. Documentation Module:

- **Responsibilities:**
    - Provide comprehensive documentation for rule definition syntax, API usage, and deployment.
- **Implementation:**
    - Develop documentation using tools like Markdown or AsciiDoc.
    - Include code comments for better code documentation.

12. Deployment Module:

- **Responsibilities:**
    - Define deployment configurations.
    - Provide instructions for deploying the Rule Engine.
- **Implementation:**
    - Create deployment scripts or Docker containers.
    - Document deployment steps and configurations.

13. Monitoring Module:

- **Responsibilities:**
  - Integrate with monitoring tools for performance tracking.
  - Generate alerts for rule execution failures.
- **Implementation:**
  - Utilize monitoring tools like Prometheus or Application Insights.
  - Implement alerting mechanisms.

14. Error Handling Module:

- **Responsibilities:**
  - Implement comprehensive error handling mechanisms.
  - Provide meaningful error messages for debugging.
- **Implementation:**
  - Use try-catch blocks for exception handling.
  - Log detailed error information.