**Rule Engine with AST**

**Objective:**

Develop a simple 3-tier rule engine application(Simple UI, API and Backend, Data) to determine

user eligibility based on attributes like age, department, income, spend etc.The system can use

Abstract Syntax Tree (AST) to represent conditional rules and allow for dynamic

creation,combination, and modification of these rules.

**Data Structure:**

● Define a data structure to represent the AST.

● The data structure should allow rule changes

● E,g One data structure could be Node with following fields

○ type: String indicating the node type ("operator" for AND/OR, "operand" for

conditions)

○ left: Reference to another Node (left child)

○ right: Reference to another Node (right child for operators)

○ value: Optional value for operand nodes (e.g., number for comparisons)

**Data Storage**

● Define the choice of database for storing the above rules and application metadata

● Define the schema with samples.

**Sample Rules:**

● rule1 = "((age > 30 AND department = 'Sales') OR (age < 25 AND

department = 'Marketing')) AND (salary > 50000 OR experience >

5)"

● rule2 = "((age > 30 AND department = 'Marketing')) AND (salary >

20000 OR experience > 5)"

**API Design:**

1. **create\_rule(rule\_string):** This function takes a string representing a rule (as

shown in the examples) and returns a Node object representing the corresponding AST.

2. **combine\_rules(rules):** This function takes a list of rule strings and combines them

into a single AST. It should consider efficiency and minimize redundant checks. You can

explore different strategies (e.g., most frequent operator heuristic). The function should

return the root node of the combined AST.3. **evaluate\_rule(JSON data):** This function takes a JSON representing the combined

rule's AST and a dictionary data containing attributes (e.g., data = {"age": 35,

"department": "Sales", "salary": 60000, "experience": 3}). The

function should evaluate the rule against the provided data and return True if the user is

of that cohort based on the rule, False otherwise.

**Test Cases:**

1. Create individual rules from the examples using create\_rule and verify their AST

representation.

2. Combine the example rules using combine\_rules and ensure the resulting AST

reflects the combined logic.

3. Implement sample JSON data and test evaluate\_rule for different scenarios.

4. Explore combining additional rules and test the functionality.

**Bonus:**

● Implement error handling for invalid rule strings or data formats (e.g., missing operators,

invalid comparisons).

● Implement validations for attributes to be part of a catalog.

● Allow for modification of existing rules using additional functionalities within

create\_rule or separate functions.This could involve changing operators, operand

values, or adding/removing sub-expressions within the AST.

● Consider extending the system to support user-defined functions within the rule

language for advanced conditions (outside the scope of this exercise).