# APPLICATION 1: RULE_ENGINE_WITH_AST

This project implements a rule engine using Abstract Syntax Tree (AST) to dynamically create, combine, and evaluate rules based on user inputs. It features an interactive frontend where users can define rules, combine multiple rules, and evaluate them against custom datasets. The rules are stored in a MongoDB database, and the backend provides RESTful APIs for creating and managing the rules.

## ## Features

**Create Rule: Users can define rules with conditional logic (e.g., `age > 25 AND income >= 50000`).

**Combine Rules: Multiple rules can be combined using logical operators (`AND`, `OR`).

**Evaluate Rules: Evaluate any created rule against a data set provided by the user.

**Persistent Storage: Rules are stored in MongoDB for future retrieval and evaluation.

## ## Project Structure

**Backend Folder:

```
backend
│── ruleController.js     # Handles rule creation, combination, and evaluation
│── Rule.js               # Mongoose model for rules stored in MongoDB
│── ruleRoutes.js         # Defines API routes for rule operations
│── ast.js                # Utilities for AST operations
├── server.js             # Main server file to run the backend
```

**Frontend Folder:

```
frontend
│── index.html
├── server.js           # Frontend for creating, combining, and evaluating rules
```

**### `ruleController.js`**

This file will contain the logic for creating rules, combining them, and evaluating them.

**### `Rule.js`**

Defines the schema for storing the rules in MongoDB.

**### `ruleRoutes.js`**

Contains the route definitions for handling requests related to rule operations (e.g., create, combine, evaluate).

**### `ast.js`**

This file contains helper functions for parsing and evaluating rules using AST.

**### `index.html`**

The frontend will provide a user interface to interact with the rule engine, allowing users to create and evaluate rules.

**### `server.js`**

This is the main file to start the Node.js backend server, connecting the controller, routes, and database.

**## How It Works**

**### Rule Creation**

The rule engine allows users to define custom rules in the form of strings. These rules are parsed into an Abstract Syntax Tree (AST) and stored in MongoDB for future use.

Example rule: `age > 25 AND income >= 50000`

**### Combining Rules**

Users can combine multiple rules with logical operators (`AND` or `OR`). The combined rule is saved as a new rule in the database.

Example combination: Combine `rule1` and `rule2` using `AND`.

**### Rule Evaluation**

The engine can evaluate a stored rule against custom data. The data is passed as JSON, and the rule logic is applied to determine if the condition is met.

Example data: `{ "age": 30, "income": 60000 }`

**## API Endpoints**

- `POST /api/rules/create_rule`: Creates a new rule from a string.

  - **Request Body**:
    ```json
    {
      "ruleName": "myRule",
      "ruleString": "age > 25 AND income >= 50000"
    }
    ```

- `POST /api/rules/combine_rules`: Combines two or more rules with a logical operator.
  - **Request Body**:
    ```json
    {
      "rules": ["rule1", "rule2"],
      "op": "AND"
    }
    ```

- `POST /api/rules/evaluate_rule`: Evaluates a rule against a set of data.
  - **Request Body**:
    ```json
    {
      "ast": "myRule",
      "data": { "age": 30, "income": 60000 }
    }
    ```

## Installation

**1. Clone the repository:**

   ```bash
   git clone https://github.com/jamedarkhasim/ Rule_Engine_with_AST.git
   ```

**2. Navigate to the project directory:**

```bash
cd Rule_Engine_with_AST
```

**3. Install dependencies:**

```bash
npm install
```

**4. Set up the MongoDB** connection in `server.js` (update the connection string if necessary).

**5. Start the server:**

```bash
npm start
```

6. Open your browser and navigate to **`http://localhost:8000`.**

## ## Frontend (HTML Form)

The frontend is a simple HTML form (`index.html`) that allows users to interact with the rule engine. Users can create rules, combine them, and evaluate rules directly from the UI.

### ### Create Rule

- Enter a rule name and a rule string in the form and click **Create Rule**.

### ### Combine Rules

- Enter the rule names to combine, choose a logical operator (AND/OR), and click **Combine Rules**.

### ### Evaluate Rule

- Enter the rule name and the JSON data to evaluate the rule against, and click **Evaluate Rule**.

## ## Technologies Used

****Node.js** for the backend server

****Express.js** for routing and API endpoints

**MongoDB for persistent rule storage

**Mongoose for interacting with MongoDB

**JavaScript for both frontend and backend logic

**HTML/CSS for the frontend interface

## Future Enhancements

Add user authentication and authorization.

Improve error handling and validation.

Implement rule editing and deletion features.

Extend the rule engine to support more complex expressions and operators.

## Contact

**Author: Jamedar Mohammed Khasim

**Email: jamedarmdkhasim@gmail.com

**Repository link: https://github.com/jamedarkhasim/Rule_Engine_with_AST.git