rule-engine-ast

Flask Application for Rule Evaluation

Overview

This Flask application evaluates business rules based on provided criteria, allowing for dynamic rule creation and evaluation. Users can create rules, evaluate them against data, and combine multiple rules into a single rule for more complex evaluations.

Table of Contents

Features

- Create dynamic rules using a simple API.
- Evaluate rules against input data.
- Combine multiple rules into a single rule.
- Store rules persistently in a SQLite database.

Design Choices

- *Flask Framework*: Chosen for its simplicity and ease of use for building RESTful APIs.
- *SQLite Database*: Lightweight database that simplifies setup and requires no additional configuration.
- *Rule Evaluation Logic*: Utilizes Python's eval function to dynamically evaluate rules, with necessary precautions to prevent security vulnerabilities.

Dependencies

The application requires the following dependencies to be installed:

- *Python 3.x*
- *Flask*: For building the web API.

- *SQLite*: For the database.

Getting Started

Prerequisites

- Ensure that Python 3.x is installed on your machine. You can download it from python.org.

Installation

1. Clone the repository:

   ```
   git clone https://github.com/YOUR_USERNAME/YOUR_REPOSITORY_NAME.git

    2. Navigate to the project directory:
   ```

cd YOUR_REPOSITORY_NAME

```
3.  Install required packages using pip:
```

pip install Flask

Running the Application

```
1.  Initialize the database and run the application:
```

python app.py

```
2.  Use Postman or any API client to interact with the endpoints.
```

API Endpoints

```
•   POST /api/create-rule: Create a new rule.
•   Request Body:
```

{ "rule": "(age > 30 AND department == 'Sales') OR (experience > 5)" }

```
•   POST /api/evaluate-rule: Evaluate an existing rule.
•   Request Body:
```

{ "ruleId": 1, "data": { "age": 35, "department": "Sales", "salary": 60000, "experience": 3 } }

```
•   POST /api/combine-rules: Combine multiple rules into one.
•   Request Body:
```

{ "ruleIds": [1, 2] }

Example Usage

```
1.  Creating a Rule:
```

Use the /api/create-rule endpoint to create a new rule. 2. Evaluating a Rule: After creating a rule, use the /api/evaluate-rule endpoint with appropriate data to see if it evaluates to true or false. 3. Combining Rules: Use the /api/combine-rules endpoint to combine multiple existing rules into a single new rule.

Security and Performance Considerations

```
•   Basic error handling is implemented for invalid requests.
•   Input validation and sanitization should be considered for security.
•   To improve performance, consider caching frequently accessed rules.
```

Contributing

Contributions are welcome! Please submit a pull request for any enhancements or bug fixes.

License

GitHub Link

This project is available at: https://github.com/Deeksha0307/rule-engine-ast

Instructions for Customization

- Replace YOUR_USERNAME and YOUR_REPOSITORY_NAME with your actual GitHub username and the name of your repository.

Final Steps

- Save this content in your README.md file within your project directory.
- Commit and push the changes to your GitHub repository.
- Ensure your application runs correctly, and all API endpoints work as expected.