**RULE ENGINE WITH AST**

Here I pasted all the files and folder along with codes, why because whenever I tried to give any data’s in Create Rule and Evaluate Rule, I was not taken anything in the fields but I showing only error here:

If you changed anything let me know what you changed and give it back as proper functional application to me

**backend/app.py:**

from flask import Flask, request, jsonify

import mysql.connector

app = Flask(\_\_name\_\_)

# MySQL connection setup

def get\_db\_connection():

    conn = mysql.connector.connect(

        host='localhost',  # Replace with your DB host

        user='root',       # Replace with your MySQL username

        password='#Proteus@0416',  # Replace with your MySQL password

        database='rule\_engine'

    )

    return conn

# API to create rule and store AST in MySQL

@app.route('/create\_rule', methods=['POST'])

def create\_rule():

    rule\_string = request.json.get('rule')

    try:

        ast = create\_ast(rule\_string)  # Generate AST from rule

        conn = get\_db\_connection()

        cursor = conn.cursor()

        cursor.execute("INSERT INTO rules (rule\_string, ast) VALUES (%s, %s)", (rule\_string, str(ast)))

        conn.commit()

        cursor.close()

        conn.close()

        return jsonify({"message": "Rule created successfully"}), 200

    except Exception as e:

        return jsonify({"error": str(e)}), 400

@app.route('/evaluate\_rule', methods=['POST'])

def evaluate\_rule():

    data = request.json

    rule\_id = data.get('rule\_id', 1)  # Get the rule ID from the request

    print(f"Received rule\_id: {rule\_id}")  # Debugging statement

    conn = get\_db\_connection()

    cursor = conn.cursor()

    cursor.execute("SELECT ast FROM rules WHERE id=%s", (rule\_id,))

    rule\_ast = cursor.fetchone()

    print(f"Fetched rule\_ast: {rule\_ast}")  # Debugging statement

    if rule\_ast:

        rule\_ast = rule\_ast[0]

        cursor.close()

        conn.close()

        result = evaluate\_ast(rule\_ast, data['data'])  # Pass the nested 'data' dictionary for evaluation

        return jsonify({"result": result}), 200

    else:

        cursor.close()

        conn.close()

        return jsonify({"error": "Rule not found"}), 404

def create\_ast(rule\_string):

    # For now, we will create a simple AST based on the logical operators

    # This is a placeholder; a real implementation would need a proper parser

    if 'AND' in rule\_string:

        parts = rule\_string.split('AND')

        return {

            "type": "AND",

            "left": create\_ast(parts[0].strip()),

            "right": create\_ast(' AND '.join(parts[1:]).strip()) if len(parts) > 1 else None

        }

    elif 'OR' in rule\_string:

        parts = rule\_string.split('OR')

        return {

            "type": "OR",

            "left": create\_ast(parts[0].strip()),

            "right": create\_ast(' OR '.join(parts[1:]).strip()) if len(parts) > 1 else None

        }

    else:

        # If we reach here, it's a base operand

        return {"type": "operand", "value": rule\_string.strip()}

def evaluate\_ast(ast, data):

    try:

        if ast['type'] == 'AND':

            left\_result = evaluate\_ast(ast['left'], data)

            right\_result = evaluate\_ast(ast['right'], data)

            return left\_result and right\_result

        elif ast['type'] == 'OR':

            left\_result = evaluate\_ast(ast['left'], data)

            right\_result = evaluate\_ast(ast['right'], data)

            return left\_result or right\_result

        elif ast['type'] == 'operand':

            # Evaluate the simple expressions, like "age > 30"

            expression = ast['value']

            # Replace variable names with corresponding data dictionary values

            for key, value in data.items():

                # Check if the key exists in the expression and replace it

                expression = expression.replace(key, str(value))

            # Now eval can evaluate a direct comparison

            return eval(expression)

    except Exception as e:

        print(f"Evaluation error: {str(e)}")  # For debugging purposes

        return True

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True, port=5000)

**backend/rule\_engine.py:**

class Node:

    def \_\_init\_\_(self, type, left=None, right=None, value=None):

        self.type = type

        self.left = left

        self.right = right

        self.value = value

def create\_ast(rule\_string):

    # Implement AST creation logic here

    pass

def evaluate\_ast(ast, data):

    # Implement AST evaluation logic here

    pass

**frontend/public/index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>React App</title>

</head>

<body>

    <div id="root"></div>

</body>

</html>

**frontend/src/components/RuleEvaluation.js:**

import React, { useState } from 'react';

import ruleService from '../services/ruleServices.js';

const RuleEvaluation = () => {

  const [data, setData] = useState({ age: '', department: '', salary: '', experience: '' });

  const [result, setResult] = useState(null);

  const [loading, setLoading] = useState(false);  // Loading state

  const handleSubmit = async (e) => {

    e.preventDefault();

    setLoading(true);  // Set loading to true

    try {

      const response = await ruleService.evaluateRule(data);

      setResult(response.result ? 'Eligible' : 'Not Eligible');

    } catch (error) {

      setResult('Error evaluating rule');

    } finally {

      setLoading(false);  // Set loading to false

    }

  };

  return (

    <div className="evaluate-rule-container">

        <h2 className="form-heading" style={{ color: 'white' }}>Evaluate Rule</h2>

        <form className="form-container" onSubmit={handleSubmit}>

            <div className="form-row">

            <label className="form-label" style={{ color: 'white' }}>Age:</label>

            <input className="form-input" type="number" value={data.age} onChange={(e) => setData({ ...data, age: e.target.value })} required />

            </div>

            <div className="form-row">

            <label className="form-label" style={{ color: 'white' }}>Department:</label>

            <input className="form-input" type="text" value={data.department} onChange={(e) => setData({ ...data, department: e.target.value })} required />

            </div>

            <div className="form-row">

            <label className="form-label" style={{ color: 'white' }}>Salary:</label>

            <input className="form-input" type="number" value={data.salary} onChange={(e) => setData({ ...data, salary: e.target.value })} required />

            </div>

            <div className="form-row">

            <label className="form-label" style={{ color: 'white' }}>Experience:</label>

            <input className="form-input" type="number" value={data.experience} onChange={(e) => setData({ ...data, experience: e.target.value })} required />

            </div>

            <button type="submit" className="form-button" disabled={loading} >

            {loading ? 'Evaluating...' : 'Evaluate Rule'}

            </button>

        </form>

        {result && <p>Result: {result}</p>}

    </div>

  );

};

export default RuleEvaluation;

**frontend/src/components/RuleForm.js:**

import React, { useState } from 'react';

import axios from 'axios';

import { toast, ToastContainer } from 'react-toastify';

import 'react-toastify/dist/ReactToastify.css';

const RuleForm = () => {

  const [rule, setRule] = useState('');

  const [message, setMessage] = useState('');

  const [loading, setLoading] = useState(false);  // Loading state

  const createRule = async (ruleString) => {

    try {

      const response = await axios.post('http://localhost:5000/create\_rule', { rule: ruleString });

      return response.data.message;

    } catch (error) {

      console.error("Error Creating Rule:", error);

      // Show custom alert on error

      toast.error('Invalid input: Please enter a valid rule', {

        position: 'top-right'  // Corrected position reference

      });

      throw new Error('Error Creating Rule');

    }

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    setLoading(true);  // Set loading to true

    try {

      const successMessage = await createRule(rule);

      setMessage(successMessage);

      // Optionally, show success notification

      toast.success('Rule created successfully!', {

        position: 'top-right'  // Corrected position reference

      });

    } catch (error) {

      setMessage(error.message);

    } finally {

      setLoading(false);  // Set loading to false

    }

  };

  return (

    <div>

      <ToastContainer /> {/\* Container for toast notifications \*/}

      <h2 className="form-heading">Create Rule</h2>

      <form className='form-container' onSubmit={handleSubmit}>

        <label className="form-label">Rule:</label>

        {/\* <input

          className="form-input"

          type="text"

          value={rule}

          onChange={(e) => setRule(e.target.value)}

          placeholder="Enter rule (e.g., age > 30 AND salary > 50000)"

          required

        /> \*/}

        <input

          className="form-input"

          type="text"

          value={rule}

          onChange={(e) => {

              setRule(e.target.value);

              console.log(e.target.value); // Debugging to see what value is being entered

          }}

          placeholder="Enter rule (e.g., age > 30 AND salary > 50000)"

          required

        />

        <button type="submit" className="form-button" disabled={loading}>

          {loading ? 'Creating...' : 'Create Rule'}

        </button>

      </form>

      {message && <p>{message}</p>}

    </div>

  );

};

export default RuleForm;

**frontend/src/services/ruleServices.js:**

import axios from 'axios';

const API\_URL = 'http://localhost:5000'; // Flask API

const ruleService = {

    createRule: async (ruleString) => {

        const response = await axios.post(`${API\_URL}/create\_rule`, { rule: ruleString });

        return response.data;

    },

    evaluateRule: async (data) => {

        const response = await axios.post(`${API\_URL}/evaluate\_rule`, data);

        return response.data;

    }

};

export default ruleService;

**frontend\src\App.js:**

import React from 'react';

import RuleForm from './components/RuleForm';  // Import RuleForm component

import RuleEvaluation from './components/RuleEvaluation';  // Import RuleEvaluation component

import './App.css';  // Import the CSS file

function App() {

  return (

    <div className="App" style={{

      backgroundImage: `url(/backgroundimage.png)`,  // Reference the image in the public folder

      backgroundSize: 'cover',

      backgroundPosition: 'center',

      backgroundRepeat: 'no-repeat',

      height: '100vh',

      width: '100vw',

      margin: 0,

      padding: 0,

      display: 'flex',

      flexDirection: 'column',

      justifyContent: 'flex-start',

      alignItems: 'center',

      textAlign: 'center',

    }}>

      <h1 style={{ color: 'white', marginTop: '150px'}}>Rule Engine with AST</h1>

      <RuleForm />

      <RuleEvaluation />

    </div>

  );

}

export default App;

**frontend\src\App.css:**

/\* RuleForm.js \*/

.form-container {

  display: flex; /\* Use flexbox for horizontal layout \*/

  flex-direction: row; /\* Arrange elements horizontally \*/

  align-items: center; /\* Align elements vertically \*/

  gap: 10px;

}

.form-heading {

  color: white;

}

.form-label {

  display: flex;

  flex: 0 0 auto; /\* Set a fixed width for the label \*/

  margin-right: 10px; /\* Add spacing between label and input \*/

  color: white;

  padding-right: 5px;

  font-size: larger;

  margin-bottom: 10px;

}

.form-input {

  flex: 1 1 auto; /\* Allow the input to grow or shrink \*/

  padding: 8px;

  border-radius: 10px;

  max-width: 400px;

  box-sizing: border-box;

}

.form-button {

  padding: 8px 20px;

  background-color: #5a67d8;

  border: none;

  border-radius: 10px;

  color: white;

  cursor: pointer;

  font-size: 1rem;

}

.form-button:disabled {

  background-color: #a0aec0;

}

**frontend\src\index.js:**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

**frontend\src\index.css:**

body {

  margin: 0;

  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',

    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',

    sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

}

code {

  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',

    monospace;

}