Calculator Challenge

Overview

This challenge involves creating a web-based calculator application that performs basic arithmetic operations. The focus is on a robust design that separates business logic from presentation, and is extendable for future enhancements.

Functional Requirements

1. **Core Operations**: The calculator must support addition (+), subtraction (-), division (/), and multiplication (*).
2. **Client-side Input Format**: It takes an operator and two operands as inputs.

3. **REST API Integration:** All operations and their results should be accessible via a REST API.
4. **Server-Side Logic**: The business logic, including calculations and result processing, must be implemented on the server side.
5. **Client-Side Presentation**: The client-side application should handle only the presentation aspects and interact with the server using AJAX/REST API.
6. **Multi-Application Support**: The REST API should be designed to accommodate multiple client applications without requiring changes to the API contract.

Design and Implementation Standards

1. **OOP/OOD Principles**: The application should adhere to Object-Oriented Programming and Design principles.
2. **Extensibility**: Design the application to be open for extensions but closed for modifications (Open/Closed Principle). This means the core code shouldn't need changes for future extensions.
3. **Future Extensions (For Design Consideration Only)**:
   - Additional Operations (e.g., exponentiation - x^y).
   - Complex Expressions (e.g., $a + b - c/d + x * d$).
   - Varied Result Formats:
     - Simple Numeric Result (e.g., `{result:5}`).
     - Numeric Result with Color Coding (e.g., `{result:5, color:"red"}`). Color determination is based on the result number's parity (even: green, odd: red).
   - The design should account for these potential extensions without implementing them at this stage.

Bonus Challenge: Implement an algorithm that organizes the operators and operands in a specific order.

Implementation Notes

- Ensure that the API and server-side logic are robust and efficient.
- The client-side should be user-friendly and interact seamlessly with the server.

This challenge is an opportunity to demonstrate your skills in software design, RESTful API development, and client-server architecture.