Polynomial Processing API

October 18, 2024

Objective

The purpose of this task is for the candidate to demonstrate their skills in API design, object-oriented programming (OOP), data structures, working with databases, and testing. You need to design and implement an API that simplifies polynomial expressions and stores queries in a database. If the same expression has been processed before, the result should be retrieved from the database cache instead of recalculating it.

Task Description

You are tasked with creating a RESTful API that performs the following:

1. **Simplify a Polynomial:**
   - The API should accept a polynomial expression as input (in a string format). For example, a valid input could be: $2 * x^2 + 3 * x - 5$.
   - The system should parse the input, simplify it by combining like terms, and return the simplified polynomial.
     - Input: $2 * x^2 + 3 * x - 5 + x^2 + x$
     - Output: $3 * x^2 + 4 * x - 5$
   - The API should handle polynomial multiplication as well.
     - Input: $(x + 2) * (x - 1)$
     - Output: $x^2 + x - 2$
2. **Evaluate the Polynomial for a Given Value of x:**
   - The API should allow users to evaluate the simplified polynomial at a given value of x.
     - Input: $3 * x^2 + 4 * x - 5$, $x = 2$
     - Output: 11
3. **Database Integration and Caching:**
   - Every polynomial expression that is simplified or evaluated should be stored in a database.
   - If a polynomial expression is submitted more than once, the system should retrieve the previously stored result from the database instead of recalculating it.
   - The database should be capable of efficiently storing and retrieving both simplified expressions and evaluation results.

Requirements

1. **Object-Oriented Design:**
   - Use object-oriented principles to design your solution. The polynomial should be represented as a class with appropriate simplification, evaluation, and multiplication.
2. **Data Storage:**
   - Implement database integration to store polynomial queries and results.
   - Use any relational database (e.g., MySQL, PostgreSQL) for storing the data. You may also use an ORM tool like Hibernate for database interaction.
3. **Validation:**
   - Implement error handling for invalid inputs, such as malformed polynomial expressions or non-numeric values for x.
   - Consider the it should be easy to add another validation by customer requirement
4. **Caching Logic:**
   - Ensure the service checks the database for previously processed expressions before performing any new calculations.
   - If an expression was processed earlier, the result should be fetched from the database and returned as a cached result.
5. **Testing:**
   - Write unit tests to verify the correctness of your polynomial processing logic.
   - Additionally, include integration tests to ensure that the API interacts correctly with the database and handles caching efficiently.

Submission

- Submit your solution as a Git repository link.
- Include instructions for setting up the project, running the tests, and starting the API.