Backend Assignment - GitHub Repository Searcher

Objective:
Develop a Spring Boot application that allows users to search for GitHub repositories using the GitHub REST API. The application should store search results in a PostgreSQL database and provide an API endpoint to retrieve stored results based on filter criteria.

Requirements:

1. **GitHub Repository Search**:
    - Implement a REST API endpoint in the Spring Boot application that fetches repository details from GitHub based on search criteria such as:
        - Repository name (partial or full match).
        - Programming language.
        - Sort by (stars, forks, or updated date).
    - Fetch the data from the GitHub API (`https://api.github.com/search/repositories`).
2. **Store Results in PostgreSQL**:
    - Save the search results (repository details) in a PostgreSQL database with the following fields along with other Relevant statistics of the respository:
        - Repository ID (from GitHub, unique).
        - Name.
        - Description.
        - Owner name.
        - Programming language.
        - Stars count.
        - Forks count.
        - Last updated date.
3. **Retrieve Stored Results**:

    - Provide an API endpoint to retrieve stored repository details based on the following optional filters:
        - Programming language.
        - Minimum stars count.
        - Sorted by stars, forks, or updated date.
4. **Expected Behavior**:
    - If a repository already exists in the database, update its details instead of duplicating.
    - Handle edge cases like invalid API responses, rate limits, and empty search results.

API Specifications

1. Search GitHub Repositories

- **Endpoint**: `POST /api/github/search`

Request Body:

```
{
  "query": "spring boot",
  "language": "Java",
  "sort": "stars"
}
```

-

Response:

```
{
  "message": "Repositories fetched and saved successfully",
  "repositories": [
    {
      "id": 123456,
      "name": "spring-boot-example",
      "description": "An example repository for Spring Boot",
      "owner": "user123",
      "language": "Java",
      "stars": 450,
      "forks": 120,
      "lastUpdated": "2024-01-01T12:00:00Z"
    },
    ...
  ]
}
```

-

2. Retrieve Stored Results

- **Endpoint**: `GET /api/github/repositories`
- **Request Parameters**:
  - `language` (optional): Filter by programming language.
  - `minStars` (optional): Minimum stars count.
  - `sort` (optional): Sort by `stars`, `forks`, or `updated` (default is `stars`).

Example Request:
```
GET /api/github/repositories?language=Java&minStars=100&sort=stars
```

-

Response:
```
{
  "repositories": [
    {
      "id": 123456,
      "name": "spring-boot-example",
      "description": "An example repository for Spring Boot",
      "owner": "user123",
      "language": "Java",
      "stars": 450,
      "forks": 120,
      "lastUpdated": "2024-01-01T12:00:00Z"
    },
    ...
  ]
}
```

Evaluation Criteria:

- Assignment code submitted to be in **Java**
- REST Compliant APIs, (No UI)
- Testable by Postman, (No UI)
- Will prefer classes properly refactored and following design patterns,
- Clarity of API documentation and how to run the project.
- Will Prefer TDD, JUnit , **Test Coverage**: Comprehensive test cases covering various equation scenarios.
- Efficient database operations (e.g., upsert for repository data).
- Code quality and structure (e.g., modular design, clear separation of concerns).
- **Error Handling**: Robust validation and error messages.