Java Spring Backend Coding Test

Background

You are tasked with developing a backend application for an e-commerce platform using Java Spring. The application should connect to a Microsoft SQL Server database to retrieve and save data. Your main task is to create an API endpoint that calculates the total order amount for users based on specific criteria.

Database Schema

The database consists of three tables:

```
User
+----+------+-----+
| id | name | age |
+----+------+-----+

Order
+----+---------+----------+
| id | user_id | datetime |
+----+---------+----------+

OrderDetail
+----+----------+-----------+-------+
| id | order_id | item_name | total |
+----+----------+-----------+-------+
```

Check the file `Question2.sql` for detailed database schema information.

Task

Develop a RESTful API endpoint that retrieves the total order amount for each user, filtered by their names and a specified date range. The endpoint should accept the following query parameters:

- `name`: A string to filter users by name (case-insensitive, partial match)
- `fromDate`: Start date of the order period (format: YYYY-MM-DD)
- `toDate`: End date of the order period (format: YYYY-MM-DD)

API Specification

- **HTTP Method**: GET
- **Endpoint**: `/user-orders`
- **Query Parameters**:
  - `name` (optional): String
  - `fromDate` (required): String (YYYY-MM-DD)
  - `toDate` (required): String (YYYY-MM-DD)

Example Request

```
GET /user-orders?name=Test&fromDate=2024-01-01&toDate=2024-03-01
```

This request should return the total order value for each user whose name contains "Test" (case-insensitive) for orders placed between January 1, 2024, and March 1, 2024.

Example Response

```
[
  {
    "userId": 1,
    "userName": "John Test",
    "totalOrderAmount": 1500.50
  },
  {
    "userId": 3,
    "userName": "Alice Tester",
    "totalOrderAmount": 750.25
  }
]
```

Requirements

1. Use Java Spring Boot to create the application.
2. Implement proper error handling and validation for input parameters if possible.
3. Implement appropriate indexing on the database tables to optimize query performance.

Sample Data

Users

```
+----+------------------+-----+
| id | name             | age |
+----+------------------+-----+
| 1  | John Doe         | 30  |
| 2  | Jane Smith       | 25  |
| 3  | Alice Johnson    | 35  |
| 4  | Bob Williams     | 28  |
| 5  | Charlie Brown    | 40  |
+----+------------------+-----+
```

Orders

```
+----+---------+---------------------+
| id | user_id | datetime            |
+----+---------+---------------------+
| 1  | 1       | 2024-01-15 10:00:00 |
| 2  | 1       | 2024-02-20 14:30:00 |
| 3  | 2       | 2024-02-10 09:15:00 |
| 4  | 2       | 2024-03-05 16:45:00 |
| 5  | 3       | 2024-01-25 11:30:00 |
| 6  | 3       | 2024-02-28 13:00:00 |
| 7  | 3       | 2024-03-10 15:45:00 |
| 8  | 4       | 2024-02-05 08:30:00 |
| 9  | 4       | 2024-03-15 12:15:00 |
| 10 | 5       | 2024-01-30 10:45:00 |
| 11 | 5       | 2024-03-20 14:00:00 |
+----+---------+---------------------+
```

Order Details

```
+----+----------+---------------+--------+
| id | order_id | item_name     | total  |
+----+----------+---------------+--------+
| 1  | 1        | Laptop        | 999.99 |
| 2  | 1        | Mouse         |  29.99 |
| 3  | 2        | Keyboard      |  89.99 |
| 4  | 2        | Monitor       | 299.99 |
| 5  | 2        | Headphones    |  79.99 |
| 6  | 3        | Smartphone    | 599.99 |
| 7  | 4        | Tablet        | 399.99 |
| 8  | 4        | Case          |  39.99 |
| 9  | 5        | Printer       | 199.99 |
| 10 | 6        | Ink Cartridge |  49.99 |
| 11 | 6        | Paper         |  19.99 |
| 12 | 7        | External HDD  | 129.99 |
| 13 | 8        | Gaming Mouse  |  69.99 |
| 14 | 8        | Mousepad      |  19.99 |
| 15 | 9        | Webcam        |  89.99 |
| 16 | 10       | Office Chair  | 249.99 |
| 17 | 11       | Desk Lamp     |  39.99 |
| 18 | 11       | USB Hub       |  29.99 |
+----+----------+---------------+--------+
```

Sample Queries and Expected Outputs

Query 1: Total order amount for users with names containing "John" between 2024-01-01 and 2024-03-01

```
GET /user-orders?name=John&fromDate=2024-01-01&toDate=2024-03-01
```

```
Expected Output:
[
  {
    "userId": 1,
    "userName": "John Doe",
    "totalOrderAmount": 1499.95
  },
  {
    "userId": 3,
    "userName": "Alice Johnson",
    "totalOrderAmount": 269.97
  }
]
```

Query 2: Total order amount for all users between 2024-02-01 and 2024-03-15

```
GET /user-orders?fromDate=2024-02-01&toDate=2024-03-15

Expected Output:
[
  {
    "userId": 1,
    "userName": "John Doe",
    "totalOrderAmount": 469.97
  },
  {
    "userId": 2,
    "userName": "Jane Smith",
    "totalOrderAmount": 1039.97
  },
  {
    "userId": 3,
    "userName": "Alice Johnson",
    "totalOrderAmount": 199.97
  },
  {
    "userId": 4,
    "userName": "Bob Williams",
    "totalOrderAmount": 179.97
  },
  {
    "userId": 5,
    "userName": "Charlie Brown",
    "totalOrderAmount": 0.00
  }
]
```

Query 3: Total order amount for users with names containing "i" between 2024-01-01 and 2024-03-31

```
GET /user-orders?name=i&fromDate=2024-01-01&toDate=2024-03-31

Expected Output:
[
  {
    "userId": 2,
    "userName": "Jane Smith",
    "totalOrderAmount": 1039.97
  },
  {
    "userId": 3,
    "userName": "Alice Johnson",
    "totalOrderAmount": 399.96
  },
  {
    "userId": 4,
    "userName": "Bob Williams",
    "totalOrderAmount": 179.97
  },
  {
    "userId": 5,
    "userName": "Charlie Brown",
    "totalOrderAmount": 319.97
  }
]
```

Good luck with your implementation!