Sales Order Technical Test with Angular and SQLite

Instructions

This repository contains a technical test designed to manage sales orders using Angular, JavaScript, and SQLite.

The test requirements have been adapted to replace the original .NET technology stack with Angular for the frontend

and SQLite as the database.

Your task is to create a web application with the following specifications:

1. Sales Order API

Create a REST API that can handle creating, updating, deleting, and viewing sales orders and order line information.

- The backend data store should use SQLite.
- The data model should include Orders and associated Order Lines (as shown in Expected DBO Model).
- The API should convert a received Shipment Order Request into the DBO model before storing it.

Expected DBO Model:

```
{
    "order": {
```

```
        "orderRef": "TestSO123",
        "orderDate": "2023/01/23",
        "currency": "EUR",
        "shipDate": "2023/01/23",
        "categoryCode": "B2C",
        "lines": [
                {
                    "sku": "TestSKU1",
                    "qty": 1,
                    "desc": "Test SKU 1"
                },
                {
                    "sku": "TestSKU2",
                    "qty": 4,
                    "desc": "Test SKU 2"
                }
            ]
    }
}
```


Shipment Order Request:


```
{
    "SalesOrderRequest": {
        "SalesOrder": {
```

```
        "salesOrderRef": "TestSO123",
        "orderDate": "2023/01/23",
        "currency": "EUR",
        "shipDate": "2023/01/23",
        "categoryCode": "B2B",
        "orderLines": [
            {
                "skuCode": "TestSKU1",
                "quantity": 1,
                "description": "Test SKU 1"
            },
            {
                "skuCode": "TestSKU2",
                "quantity": 4,
                "description": "Test SKU 2"
            }
        ]
    }
  }
}
```


2. Frontend Application

Develop an Angular frontend application that allows users to manage sales orders using the API.

- Create a form to capture or update sales order information:
    a) Sales Order Reference (Textbox)
    b) Currency (Textbox)
    c) Order Date (Date Picker)
    d) Category Code (Dropdown: B2B, B2C)
    e) Ship Date (Current Date/Time)
    f) Order Lines with fields for SKU Code, Product Description, and Quantity.

- Create a view to list all orders (only header details), with options to search by Order Reference, Order Type, and Create Date (From and To).
- When an order is selected, display the Order Lines below the header list, allowing for inline editing or deletion of order lines. Ensure edits are saved back to the database.

3. Role-based Access Control

Implement authentication and role-based access control using Passport.js and SQLite.

- Implement two roles:
    a) Admin: Full CRUD access to all resources.
    b) Guest: Read-only access to order details.

- Include an authentication system with a login page that allows users to log in with username and

password.

- Use Passport.js strategies to control access to the various API endpoints based on the user's role.

4. Additional Instructions

- Store the solution files in an organized structure and ensure SQLite scripts are provided for initializing

the database.

- After completing the test, package the solution as a zip file and include necessary setup instructions.