**Programming Assignment:**

Create a Spring Boot project that provides the REST endpoints for a basic inventory management system. For this project, the inventory system contains only a single entity named Item.

Each item of data is a JSON object. Each such object has the following properties:

* itemId: the unique integer ID of the object
* itemName: the name of the item
* itemEnteredByUser: the name of the user entering the item
* itemEnteredDate: the timestamp at which the item is saved in the database (defaults to the current system timestamp)
* itemBuyingPrice: a Double value, up to one decimal place, denoting the item's buying price
* itemSellingPrice: a Double value, up to one decimal place, denoting the item's selling price
* itemLastModifiedDate: the timestamp at which the item is modified in the database (defaults to the current system timestamp)
* itemLastModifiedByUser: the name of the user modifying the item
* itemStatus: enum value, either AVAILABLE or SOLD

Here is an example of an item data JSON object:

{

"itemId":1,

"itemName":"item\_x",

"itemEnteredByUser":"user\_x",

"itemEnteredDate":"2020-05-10T13:00:41.499",

"itemBuyingPrice":50.0,

"itemSellingPrice":55.0,

"itemLastModifiedDate":"2020-05-10T13:00:41.498",

"itemLastModifiedByUser":"user\_y",

"itemStatus":"AVAILABLE"

}

Implement the /app/item REST endpoint for the following 8 operations:

POST request to /app/item :

* should accept POST requests at /app/item and item data as a JSON body
* if the itemId exists in the database, then it should return status code 400
* If the itemId doesn't exist in the database, then it should insert the data and return the inserted item as a response with status code 201

PUT request to /app/item/{itemId}:

* should accept PUT requests at /app/item/{itemId} and item data as a JSON body, where itemId is a path variable
* if the itemId exists in the database then it should update and return the updated item as a response with status code 200
* if the itemId doesn't exist in the database, it should return status code 404

DELETE request to /app/item/{itemId}:

* should accept DELETE requests at /app/item/{itemId} where itemId is a path variable
* if the itemId exists in the database, then it should delete the specified item and return status code 200
* if the itemId doesn't exist in the database, it should return status code 400

DELETE request to /app/item:

* should accept DELETE requests at /app/item
* should delete all the items from the database and return status code 200

GET request to /app/item/{itemId}:

* should accept GET requests at /app/item/{itemId} where itemId is a path variable
* if the itemId exists in the database, then it should return the item with status code 200
* if the itemId doesn't exist in the database, it should return status code 404

GET request to /app/item:

* should accept GET requests at /app/item
* should return all the items from the database with return status code 200

GET request to /app/item?itemStatus={status}&itemEnteredByUser={enteredBy}:

* should accept GET requests at /app/item?itemStatus={status}&itemEnteredByUser={enteredBy}
* should return all the items having itemStatus=status and itemEnteredByUser=enteredBy, where status and enteredBy are request params, with status code 200

GET request to /app/item?pageSize={pageSize}&page={page}&sortBy={sortByField}:

* should accept GET requests at /app/item?pageSize={pageSize}&page={page}&sortBy={sortByField}
* should return the requested page by paginating with pageSize and sorting by the sortBy field

1. Implement the above 8 endpoints in the ItemController class, and the corresponding service methods in the ItemService class. Implement the POST request to /app/item first because testing the other methods requires POST to work correctly.
2. Protect these API endpoints via OpenID Connect/SAML/OAuth2.0. Use any free IDP providers like OKTA, Auth0, or Google if you can.
3. A Single Page App (SPA) to consume all the API endpoints through ReactJS/AngularJS components [screenshots below]
4. Deploy this application locally on your laptop OR with any cloud provider of your choice for a quick demo.

**Screenshots for the UI:**

**Table

Description automatically generated**

**Table

Description automatically generated**