**GROCERY API**

# Problem Statement :

* **Design a Grocery Booking API:**
* Roles:

a) Admin

b) User

* **Design API endpoints:**
  + - Admin Responsibilities:

- Add new grocery items to the system

- View existing grocery items

- Remove grocery items from the system

- Update details (e.g., name, price) of existing grocery items

- Manage inventory levels of grocery items

* + - User Responsibilities:

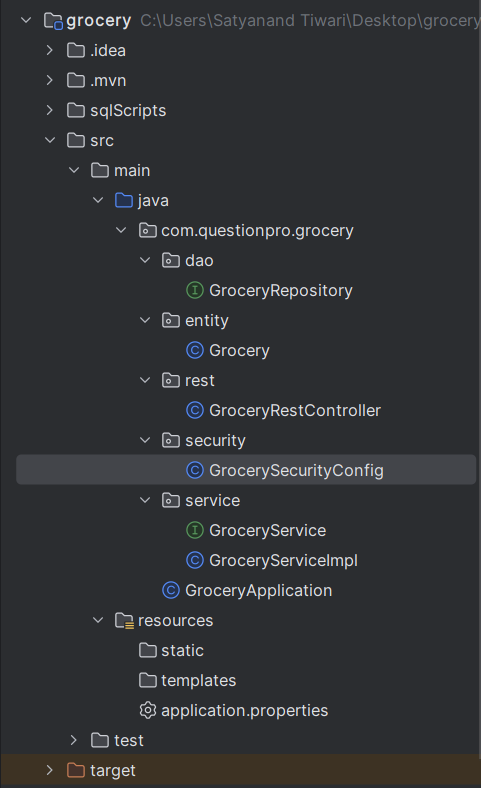
- View the list of available grocery items

- Ability to book multiple grocery items in a single order

* **Advanced Challenge:**
* Containerize the application using Docker for ease of deployment and scaling.
* Database: Use any relational database of your choice.

# API Design :

1. **DAO layer:** Consists of GroceryRepository.
2. **Entity package:** Contains entity class i.e., Grocery. Blueprint for database table.
3. **Rest layer:** GroceryRestController present in this package. Call from endpoints of Grocery API received here.
4. **Security package:** Defines security configuration and exposes endpoints based on roles.
5. **Service layer:** Contains service class i.e., GroceryServiceImpl and its interface GroceryService.



*Fig:* Packages defined in the Grocery API

* API Implementation :

Created two users for the API. The first user has name “user” and has the role of user. The second user has name “admin” with role of admin.

USERS:

1. Name: user

Password: hi

1. Name: admin

Password: hello

ACCESSIBILITY OF ENDPOINTS:

1. **User:**
   * View: Can view list of groceries in the database and a single item in database using the id of grocery item.
     + <http://localhost:8080/api/groceries> {GET}
     + [http://localhost:8080/api/groceries/{groceryId}](http://localhost:8080/api/groceries/%7bgroceryId%7d) {GET}
   * Create List: Can create a list of grocery items to buy the items in the database table. The request will contain the list of grocery items and its count. The response would consist of a list with items requested by user. If the count requested by user is more than the count of grocery item in the database, the count returned would be either the amount present in database table or 0 if database has no item.
     + <http://localhost:8080/api/groceries/buy> {PUT}
2. **Admin:**
   * View: Can view list of groceries in the database and a single item in database using the id of grocery item.
     + <http://localhost:8080/api/groceries> {GET}
     + [http://localhost:8080/api/groceries/{groceryId}](http://localhost:8080/api/groceries/%7bgroceryId%7d) {GET}
   * Add: Can add any grocery item to the database. The request will consist of json object containing columns in database table i.e., name, price, count. The response will contain json object with the item added in database table along with the id of item.
     + <http://localhost:8080/api/groceries> {POST}
   * Update: Can update grocery in the database. Admin can update any of the column in database table i.e., name, price, count thus also helping in maintaining inventory level. The request will contain the parameters to change in database in form of json. The response returned would be json object updated in database.
     + <http://localhost:8080/api/groceries> {PUT}
   * Delete: Can delete any grocery item from the database.
     + [http://localhost:8080/api/groceries/{groceryId}](http://localhost:8080/api/groceries/%7bgroceryId%7d) {DELETE}

* Advanced Challenge:

Use of **MySQL** database to store the grocery items. The scripts for the database table are provided in the folder ***sqlScripts.***

SCRIPTS:

1. **create-user:** To create a user “springgrocery” and provide it privileges
2. **grocery-directory**: Creates database “grocerydirectory” and in it creates a table called “grocery” for storing the inventory/grocery items. Scripts inserts some items in the table “grocery”.
3. **setup-spring-security-demo-database-bcrypt:** Creates tables “users” and “authorities” and defining roles for each user. Also adds password for each user in bcrypt format for additional security. (Bcrypt passwords can be created at : <https://www.bcryptcalculator.com> )

MySQL DEFINITIONS:

URL-> jdbc:mysql://localhost:3306/grocery\_directory

Username -> springgrocery

Password -> springgrocery

* Additional Features:

Added support for Swagger plugin. It can be accessed only by the admin and not by the user. It auto creates documentation for the API.

Endpoint for Swagger UI: <http://localhost:8080/swagger-ui/index.html>

