Task Intro

In this task, you will be enhancing the ShoppingBasket module developed in previous tasks by providing a REST interface.

Unlike in the previous tasks, we will not have a simple web interface. All interaction will be conducted over REST.

The base implementation of the ShoppingBasket module will be provided, and you can start the REST application by running `gradle bootRun`. You can access the current shop by running:

```
curl -s -X GET http://localhost:8080/api/shop/viewallLinks to an external site.
```

The application supports multiple users, and each user will have a shopping basket associated with that user. You will notice that we have a ShoppingBasketRepository, which persists the class ShoppingBasket, and that each ShoppingBasket has an id, user, name, and count. (These are not named consistently, or follow good practice, but are an example of what you may find in some systems.) A single ShoppingBasket object stores the number of a specific fruit on a specific user's basket. The id is the primary key used in the database uniquely identifying each row. That is, the following object

ShoppingBasket(id: 10, user: userA, name: Apple, count: 10)
ShoppingBasket(id: 11, user: userA, name: Orange, count: 1)

Represents 10 apples that are stored by userA in their basket, and 1 orange. There may be other objects such as

ShopingBasket(id: 13, user: userB, name: Apple, count: 1)

which represents apples stored in userB's shopping basket. All instances are available through the ShoppingBasketRepository which is available as a member variable `shoppingBasket` in ShoppingController.

You will notice that the ShoppingBasket does not have a cost member. The cost is stored separately as a HashMap in `costs` member of the ShoppingController, which provides the cost of any particular item. For example, apple may have a cost of 2.0. The total cost of a basket is computed using this cost hashmap when asked for. The cost of any item, if not provided, should be taken as 0.0. The cost map is not persistent.

You are not allowed to modify any files other than"src/main/java/au/sydney/soft3202/task1/ShoppingController.java". You are also only allowed to add methods to this class. Do not modify the declaration of any of the class member variables of ShoppingController or the Item class.

Required Features

1.View all users: You should implement a REST method to view all users of the system. The command will be:
curl -s -H "Content-Type: application/json" -X GET http://localhost:8080/api/shop/users
It should return a list of users as a JSON array. E.g. ["userA", "userB"] etc.

2. View all costs: You should implement a REST method to view all costs of the system. The command will be:
curl -s -H "Content-Type: application/json" -X GET http://localhost:8080/api/shop/costs
It should return a list of costs as a JSON map. E.g. {"apple": 10.1, "papaya": 1.0} etc.

3. View the basket of a specific user.
curl -s -H "Content-Type: application/json" -X GET http://localhost:8080/api/shop/users/userA
It should return a list of items in the current basket. No cost will be given. E.g.

```
[
  {
    "name": "papaya",
    "count": 100
  },
  {
    "name": "apple",
    "count": 10
  }
]
```

4.View the total cost in the basket of a specific user
curl -s -H "Content-Type: application/json" -X GET http://localhost:8080/api/shop/users/userA/total
It should return the total cost as a single number after computing the total cost of items using the costs map.

5. Allow addition of the cost of a new item
curl -s -H "Content-Type: application/json" -X POST -d '{"name":"papaya", "cost":"1.0"}' http://localhost:8080/api/shop/costs
This should add papaya to the costs table (overwriting if it exists).

6. Allow modification of the cost of an item
curl -s -H "Content-Type: application/json" -X PUT -d '{"name":"papaya", "cost":"2.0"}' http://localhost:8080/api/shop/costs/papaya
This should update the papaya cost in the map.

7. Add to the basket
curl -s -H "Content-Type: application/json" -X POST -d '{"name":"papaya", "count":"100"}' http://localhost:8080/api/shop/users/userA/add
This should add 100 papayas to the basket of userA. If papayas exist, then you have to increment the count.

8.Edit the basket

curl -s -H "Content-Type: application/json" -X PUT -d '{"count":"2"}' http://localhost:8080/api/shop/users/userA/basket/papaya
This should set the papaya count to 2 in the users basket.

9. Allow deleting the basket of a user
curl -s -H "Content-Type: application/json" -X DELETE http://localhost:8080/api/shop/users/userA
This should remove all items in the basket of a the user specified.

10. Allow deleting an item in the basket
curl -s -H "Content-Type: application/json" -X DELETE http://localhost:8080/api/shop/users/userA/basket/papaya
This should delete the papayas in the userA basket.
(Note: the previous version had `uses` rather than `users` in the URL. If you have submitted your work where at least one of the URLs work, you will get marks.)

1.