First I set up my entity framework database with all the tables I wanted included and their corresponding classes.

I then set up the templates required for basic crud operations using LINQ.
I edited the inventory default get inventories (all inventory data) to include information from both the user table and the item table. This way when I request information about all items in the inventory I can have a view of who owns which item, and the name of the item owned.

And as we can see from swagger that is working.

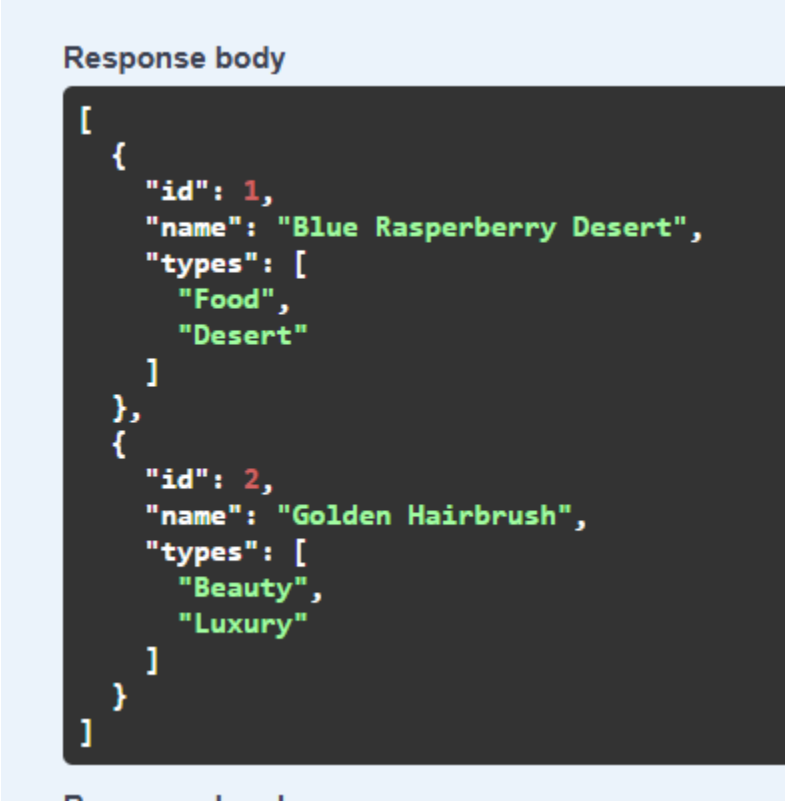
Response body


```
[
  {
    "id": 1,
    "quantity": 1,
    "userId": 2,
    "username": "mango_salad",
    "itemId": 1,
    "itemName": "Blue Rasperberry Desert"
  },
  {
    "id": 2,
    "quantity": 3,
    "userId": 3,
    "username": "Jerbear1997",
    "itemId": 2,
    "itemName": "Golden Hairbrush"
  }
]
```


I'm going to go ahead and create the same readability for the bridge table for types, so that we can see the item as well as its types.

Okay this is trickier than i thought with the many to many tables. I think the code is correct however I am getting a 500 error

Okay after spending an hour looking very closely at how the data was travelling I was able to get it to work. I also had to update migrations and fix a foreign key issue.



Note to self make a note describing how the data travels in this function.

Readability view for inventory table to see all items in a users inventory

Ok now we can get all the items in a specific user's inventory. This will be useful later when we want to display users items. The following is showing all user items with a userid of 2.
This was painful, I didnt have the proper HTTP routing and it didnt play well with the template causing a 500 error. It took me an hour to figure out. Well you live and you learn.


```
[
  {
    "id": 1,
    "quantity": 1,
    "userId": 2,
    "username": "mango_salad",
    "itemId": 1,
    "itemName": "Blue Rasperberry Desert"
  },
  {
    "id": 3,
    "quantity": 2,
    "userId": 2,
    "username": "mango_salad",
    "itemId": 2,
    "itemName": "Golden Hairbrush"
  }
]
```


Afterwards I want to create a part of the add item to inventory functionality that will also update the inventory space on the user table. If an item already exists in the table with the same item id and user id, then increase the quantity of that item. Will need to take an item id input and user id input.

First I created a function to add an item to the inventory table based off of the three needed parameters instead of an entire json with foreign keys etc.

```
// this is my inventory add, the other one wants a bunch of crazy json TOO MUCH I SAY
// POST: api/InventoriesAPI/AddToInventory
[HttpPost("AddToInventory")]
0 references
public async Task<ActionResult<Inventory>> PostInventory(int userId, int itemId, int quantity)
{
    // Create new inventory entry
    var inventoryItem = new Inventory
    {
        UserId = userId,
        ItemId = itemId,
        Quantity = quantity
    };

    // Add to the database
    _context.Inventory.Add(inventoryItem);
    await _context.SaveChangesAsync();

    // Return the created inventory entry
    return CreatedAtAction("GetInventory", new { id = inventoryItem.Id }, inventoryItem);
}
```

As we can see the api successfully adds to the inventory table

Response body


```
{
  "id": 6,
  "quantity": 2,
  "userId": 3,
  "itemId": 3,
  "user": null,
  "item": null
}
```


If we look at our user inventory for userId 3 we can see they now have 2 itemId 3's (turkey legs)


```
{
  "id": 2,
  "quantity": 3,
  "userId": 3,
  "username": "Jerbear1997",
  "itemId": 2,
  "itemName": "Golden Hairbrush"
},
{
  "id": 6,
  "quantity": 2,
  "userId": 3,
  "username": "Jerbear1997",
  "itemId": 3,
  "itemName": "Turkey Leg"
}
```

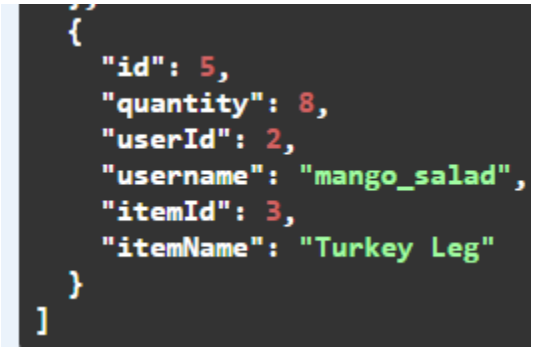

Now I will tackle changing the quantity if the item already exists in the table
Ok, the code is written i will test it now.
Below we see userId 2 (mango_salad) has 5 turkey legs. Now we will add 3 more

Response body


```
[
  {
    "id": 1,
    "quantity": 1,
    "userId": 2,
    "username": "mango_salad",
    "itemId": 1,
    "itemName": "Blue Rasperberry Desert"
  },
  {
    "id": 3,
    "quantity": 2,
    "userId": 2,
    "username": "mango_salad",
    "itemId": 2,
    "itemName": "Golden Hairbrush"
  },
  {
    "id": 5,
    "quantity": 5,
    "userId": 2,
    "username": "mango_salad",
    "itemId": 3,
    "itemName": "Turkey Leg"
  }
]
```

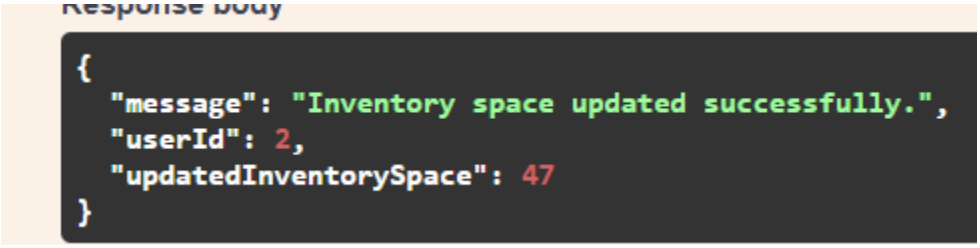

Okay that didnt work, lets try again
Changes a few incorrectly reference variables and were back in business

As we can see now, userId 2 has 3 more turkey legs than before



Ok now we have an update inventory space function in the user controller. It does not contain error handling at this time but it is working.

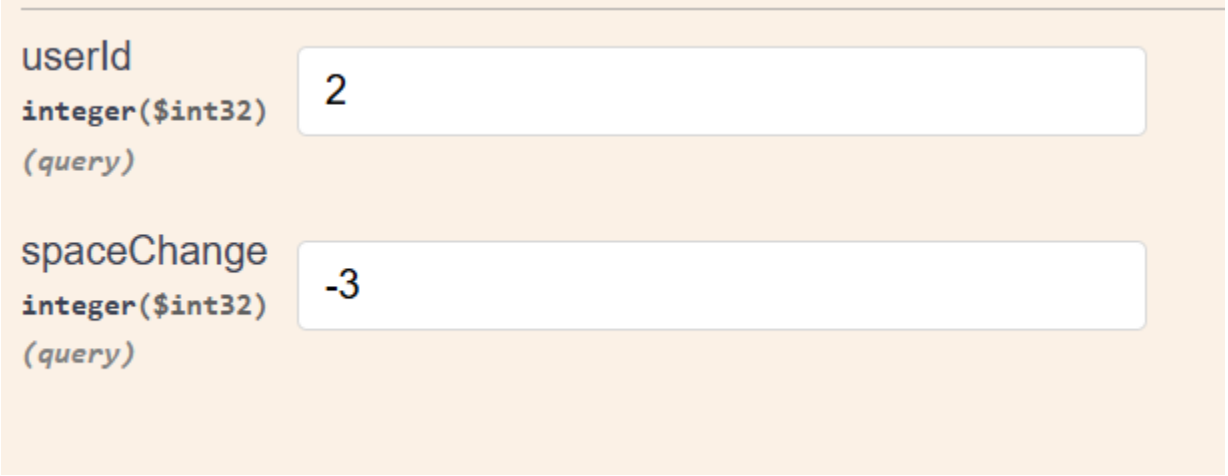
We can see here that mango_salad has an inventory space of 50, next we will use of updateInventory API to update the inventory space to 47



As we can see it was updated to 47. If we check the user again we can confirm

Now we want our add item api to call the update inventory api
Okay so I moved my update user inventory method into a user service so that both my user controller and my inventory controller can access it without having to do any crazy dependency injections or api calls between controllers.

Ok looks like it is successfully using the service



Response body


```
{
  "message": "Inventory space updated successfully.",
  "userId": 2,
  "updatedInventorySpace": 44
}
```


Now i will inject the user service into the inventory controller for use in the add item api

Ok good news is everything is being updated, however when i add a negative number it should update the inventory space to be more

Response body


```
{
  "id": 5,
  "quantity": 5,
  "userId": 2,
  "itemId": 3,
  "user": {
    "id": 2,
    "username": "mango_salad",
    "password": "mango123",
    "inventorySpace": 38,
    "solShards": 5000
  },
  "item": null
}
```


However i removed 3 items and it made inventory space go down.

Ok i changed how the math works in the user service its working fine now

Response body


```
{
  "id": 5,
  "quantity": 2,
  "userId": 2,
  "itemId": 3,
  "user": {
    "id": 2,
    "username": "mango_salad",
    "password": "mango123",
    "inventorySpace": 44,
    "solShards": 5000
  },
  "item": null
}
```


Now we will remove 1 item and see the inventory space go up

Response body


```
{
  "id": 5,
  "quantity": 1,
  "userId": 2,
  "itemId": 3,
  "user": {
    "id": 2,
    "username": "mango_salad",
    "password": "mango123",
    "inventorySpace": 45,
    "solShards": 5000
  },
  "item": null
}
```


nice

There is a lot more I would like to add, but I do need to hand this on. I will hand in what I have and continue to work on it after.

Code that will remove and item from delete table if the quantity is 0, better to check if quantity is 0? What happens in the template if an item is deleted. We need to reduce quantity not just delete the entry altogether.

Then I want to create error handling so that when an item is added to the inventory without enough space it is denied

Readme list

Methods

ListInventories()
ListUserInventory(int userId)
AddToInventory(int userId, int itemId, int quantity)
GetItemWithTypes()
updateUserSpace(int userId, int spaceRemoved)
updateUserSpace(int userId, int spaceChange)