Service Layer: Meals Weekly

While not the most interactive web app, user input is still a large part of how Meals Weekly would function. There are a few areas where the service layer is most relevant: forms for meals and adding shopping list items. They’re somewhat similar, but I’ll go over both.

**/meals**

**A screenshot of a cell phone

Description automatically generated**

Meal data will be submitted by users through this meals form.

The first service endpoint is where users input meal data (the /meals endpoint). This is one of the main features of the app, so all potential personas need to be able to input meals. It’s largely the same for each persona as well, but the form would need to be simple and visually appealing for the target audience. It would essentially consist of three text areas, one for each meal, and a submission button at the bottom. Similarly, going back to the same form would allow users to edit what they previously entered. The HTTP verbs used for this endpoint are POST and PUT.

An example request would be as follows:

{

"id": 0,

"week": "Week 1",

"day": "Sunday",

"meals: [

"breakfast": "Eggs and bacon",

"lunch": "Deli sandwich",

"dinner": "Chicken breast with rice"

]

}

The request would look the same regardless of if the user was sending a POST or PUT request. No DELETE requests are necessary because they wouldn’t be removing an entire object and could simply send a PUT request with empty strings. The most relevant error code is 405: validation exception. This would only appear if a user entered invalid data, such as data containing too many characters.

**/list**

**A screenshot of a cell phone

Description automatically generated**

Users will be able to add, update, and delete items through a shopping list UI like this one.

The “/list” endpoint would be relatively similar to the previous endpoint, except that it’s for adding list items instead of meals. Users will enter list item data, with parameters such as quantity and tags. Instead of a typical HTML form, it will be more of a CRUD interface, allowing users to select different options, including add item, edit item, and remove item. No other path parameters are necessary, since it will all be done on the /list endpoint. An example POST request for adding an item may look like this:

{

"id": 0,

"week": "Week 1",

"item": "Eggs",

"quantity": 2,

"tags: [

{

"id": 0,

"tag": "Sunday Breakfast",

}

]

}

PUT request would look largely the same and simply update the item name, quantity, or tags with new data provided by the user. This time DELETE requests would be used to remove items. Error codes would again include 405: validation exception for POST or PUT, and 404: item not found for DELETE requests.