Revised: Service Layers

Michelle Lovse

Maryville University

SWDV691

Professor Gradecki

March 27, 2021

Changes to Service Layers and Endpoints

My MVP is changing to not include the save feature of both the allergens and the searches. These two features will be moved to the stretch feature list. I will have three main categories for my endpoints that are seen below, these are user, food, and allergens. When the stretch features are incorporated, I will need to use the following for the endpoints.

For saving allergens:

PUT user/user{id}/allergen/allergen{id} where a code 200 would signal that it was created successfully.

It would look like:

```
{
        “id”: 0,
        “firstName”: “string”,
        “lastName”: “string”,
        “emailAddress”: “string”,
        “password”: “string”
        “userStatus”: 0
        {
                “Allergens”: “allergenID”,
        }
```

For saving a query it would look like:

PUT user/user{id}/food/{id}/allergen/{id} and it would have codes 200 for successful and 400 for unsuccessful save.

```
{
“id”: 0,
        “firstName”: “string”,
        “lastName”: “string”,
        “emailAddress”: “string”,
        “password”: “string”
        “userStatus”: 0
        {
                “Food”:”foodID”,
                “Allergens”: “allergenID”,
        }
```

Chart of HTTP verbs and their actions and descriptions

URI	Verb/Method	Path/Query Parameter	Description
User			
/users	GET	None	Get all users
/users/{id}	GET	User Unique Id	Get a specific user
/users?q=search term	GET	Query Search	Get a list of users with a search query or criteria
/users/{id}?fields=	GET	Unique Id, Partial Response Filters	Get a user with a filtered or partial response
/users/{id}/allergens	GET	None	Get all allergens for a user
/users/{id}/allergens/{id}	GET	Allergen Unique ID	Get a specific allergen for a users
/user	POST	User Unique ID	Create User Account
/users/{id}	DELETE	User Unique ID	Deletes User Account
User Roles			
/users/{id}/userRoles	GET	User Unique ID	Gets all privileges a user has
Allergen			

/allergen/{id}	GET	None	Get all allergens
/allergen/{id}/description	GET	Allergen Unique ID	Gets the description of the specific allergen
/allergen?q=	GET	Query Search	Query allergen cases
/allergen/{id}	DELETE	Allergen Unique ID	Deletes specific allergen
Food			
/food	GET	NONE	Gets list of all foods
/food/{id}	GET	Food Unique ID	Gets a specific food
/food/{id}/allergen/{id}	GET	Allergen Unique ID	Gets a specific allergen for a food
/food/{id}	DELETE	Food Unique ID	Deletes a specific food

POST /user

Code 200 = successful operation

Code 400 = unsuccessful operation

```
{
        “id”: 0,
        “firstName”: “string”,
        “lastName”: “string”,
        “emailAddress”: “string”,
        “password”: “string”
        “userStatus”: 0
}
```

GET /user/login
User will input their email address as a string, and their password as a string.

Code 200 = successful operation
Code 400 = Invalid email/password was entered

GET /user/logout
User will select to logout.
Code 200 = successful operation

PUT /user/{id}
Updates user information.

Code 200 = update successful
Code 400 = update failed

```
{
        “id”: 0,
        “firstName”: “string”,
        “lastName”: “string”,
        “emailAddress”: “string”,
        “password”: “string”
        “userStatus”: 0
}
```

DELETE /user/{id}

Deletes user information
Code 200 = Operation successful

Code 400 = Operation failed

POST /food

Will create a new food entry

Code 200 = successful operation
Code 400 = unsuccessful entry

```
{
        “id”: 0,
        “category”: {
                “id”:0,
                “name”: “string”
```

```
    },
    “name”: “foodName”,
    “tags”: [
    {
        “allergenID”:0,
        “name”: string
    }
    ],
}
```

PUT /food/{id}

Will update food entry

Code 200 = successful
Code 404 = food item not found
Code 403 = user not authorized

```
{
    “id”: 0,
    “category”: {
        “id”:0,
        “name”: “string”
    },
    “name”: “foodName”,
    “tags”: [
    {
        “allergenID”:0,
        “name”: string
    }
    ],
}
```

DELETE /food/{id}

Will delete food entry
code 200 = successful
code 404 = food not found
code 403 = user not authorized
code 400 = operation failed

GET allergen/

Gets list of all allergens

Code 200 = successful operation
Code 400 = unsuccessful operation

```
{
        “Allergenid”: 0,
        “name”: “string”,
        “description”: “string”,
}
```

POST /allergern/{id}

Will create new allergen listing

Code 200 = successful operation
Code 400 = unsuccessful operation

```
{
        “Allergenid”: 0,
        “name”: “string”,
        “description”: “string”,
}
```

PUT /allergen/{id}
Will update allergen record

Code 200 = successful operation
Code 400 = unsuccessful operation

```
{
        “Allergenid”: 0,
        “name”: “string”,
        “description”: “string”,
}
```

DELETE /allergen/{id}

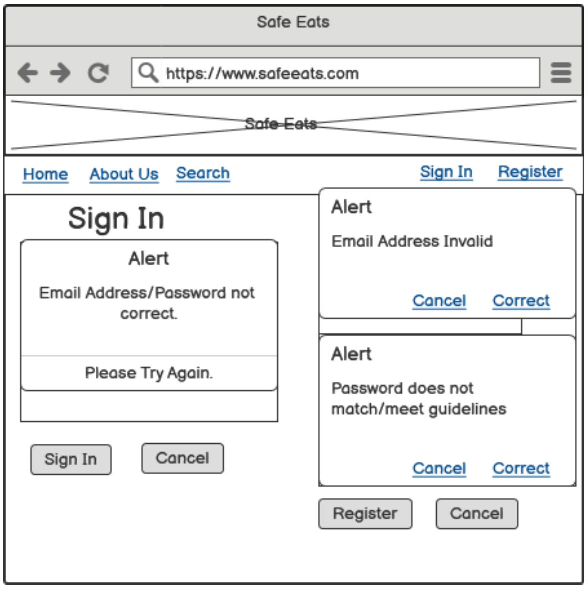
Will delete specific allergen

code 200 = successful
code 404 = allergen not found
code 403 = user not authorized
code 400 = operation failed

User Sign-In Process

On the login page, if the user is a registered user, then they will enter their credentials and submit them, which will then be verified. If the credentials are valid then the search page will be populated. If the credentials are not, then an error message will be displayed indicating that they need to re-enter their information. If the user is not registered, then they will be prompted to register for an account. The user will enter their information and the validity of the email address and the password will be verified to ensure that they meet the requirements. For instance, the user is entering an email in the proper format, and for the password, it meets password guidelines such as containing an uppercase letter, a number, and a symbol character as well as being at least 6 digits long. If the credentials do not meet these requirements, then an error message will be displayed prompting the user to correct the information. If they are correct, then the user will be signed in and the search screen will populate.

Possible Sign-In Error Messages



Alert

Email Address/Password not correct.

Please Try Again.

Alert

Email Address Invalid

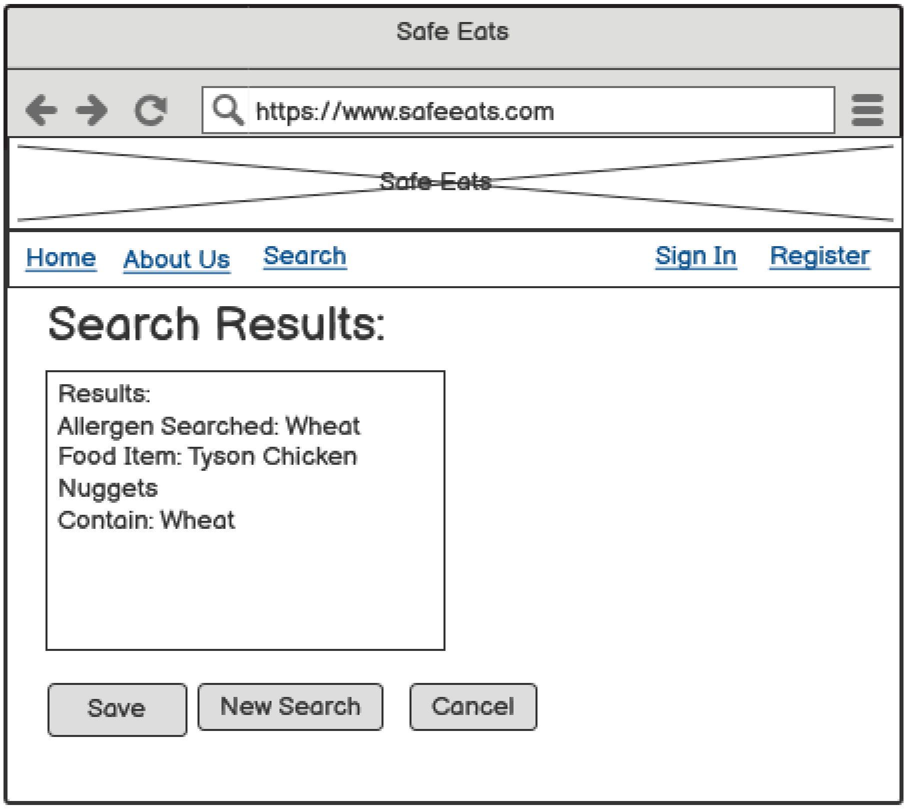
Alert

Password does not match/meet guidelines

User Search Process

The user will select the allergens through the interface by selecting the icon that correlates to their allergies. The user will then type the name of the food item that they wish to inquire about. This information will be taken by the application and form a query and return the results for the user displayed in a manner that is easy to understand.

Sample Search Results



Results:
Allergen Searched: Wheat
Food Item: Tyson Chicken Nuggets
Contain: Wheat

Figure 1. This search results feature the stretch feature of being able to save the search.

Account Types

There will be an admin account, an data entry account, and then the user accounts. The admin account will have full access to everything related to the web application. The data entry account will only have access to enter data for new foods and to make updates to anything related to the food ingredients and allergens that they contain. The data entry account will not have access to the user accounts. The user accounts will only have the ability to see their profile and saved searches, and will be able to perform searches. They will not have access to make changes to the food and allergy databases and will not have access to other user's account information.