Service Layer Design

WeEat

7/21/2024

Kayla Duffy

SWDV 691

Introduction

The backend of my web application utilizes the Express library from NodeJS. It provides a REST API from which the frontend calls to create, read, update, and delete data. The backend interacts with Firebase Firestore to store and retrieve user data, menus, and restaurant information. The backend also handles authentication through Firebase Auth and verifies tokens using middleware to protect routes. It manages user-created menus and recommended menus stored in Firebase. It also allows users to add notes to restaurant menu items and fetch menus from local restaurants. Lastly, the backend of my application utilizes a Heroku Node Proxy Server to securely handle calls to the Google Places API ensuring API keys are not exposed in the frontend.

The backend of my application is broken down into three layers:

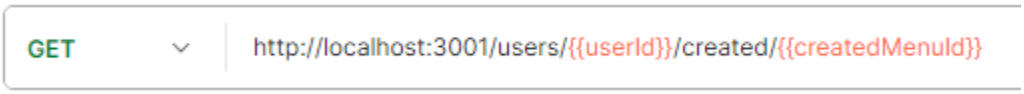
1. Routes: These define specific endpoints that the frontend can call in order to interact with the backend, mapping URL paths to corresponding functionality.
2. Controllers: These are responsible for handling incoming HTTP requests from the routes, processing them, and sending back appropriate responses.
3. Services: These contain the core business logic. Controllers delegate data processing tasks to service methods, which interact with the Firestore database to perform operations such as retrieving menus, creating menus, and deleting menus.

By breaking down the backend into these layers, it ensures a clear separation of concerns and a maintainable code structure. Each API endpoint corresponds to a specific controller function, and in turn, calls the appropriate service function to interact with Firestore.

Get Created Menus:

- Method: GET
- Endpoint: /users/{{userId}}/created/{{createdMenuId}}
- Purpose: On the user's profile, there is a button to 'View the Created Menu', which brings them to a page where the user can edit the full menu

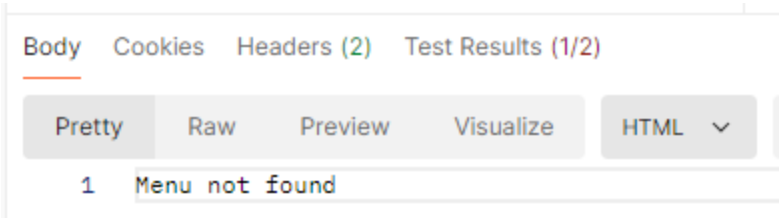
Example:



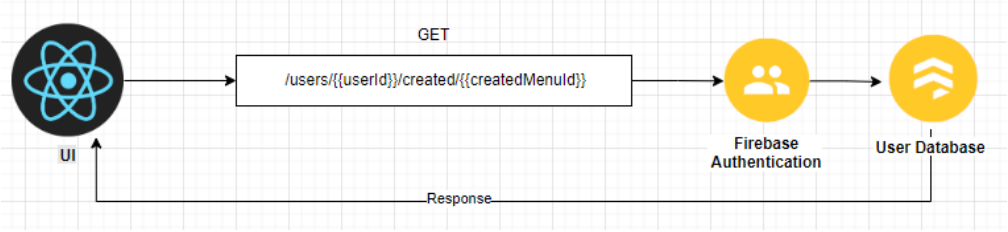
Successful Response:


```
{
    "restaurantName": "Olive Garden Italian Restaurant",
    "dishes": [
        {
            "name": "Shrimp Alfredo",
            "description": "Creamy alfredo sauce made from scratch with ingredients like parmes
            "category": "Classic Entrees",
            "allergens": [
                "Wheat",
                "Dairy",
                "Shellfish"
            ],
            "note": "This is my note",
            "id": "qMau9jvCyQiCW7UPXrRt"
        },
        {
            "note": "This is my favorite",
            "name": "Tiramisu",
            "description": "The classic Italian dessert. A layer of creamy custard set atop esp
            "category": "Desserts",
            "allergens": [
                "Dairy",
                "Egg",
                "Wheat",
                "Soy"
            ]
        }
    ]
}
```


Error Response:



Diagram



Get Personalized Menus - REVISED:

- Method: GET
- Endpoint: /users/{{userId}}/menus
- Purpose: When a user logs in and navigates to the 'Personalized Menus' page, they are met with menus that they've saved from restaurants and menus they've created.

Example Request:

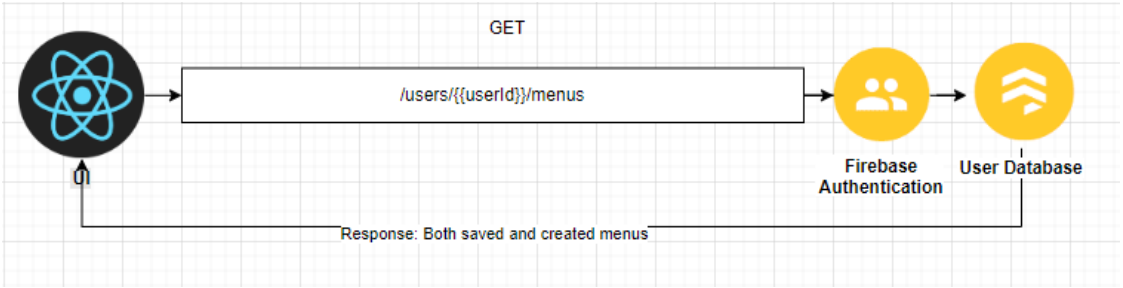
Successful Response


```
{
    "createdMenus": [
        {
            "restaurantName": "Taco Bell",
            "name": "Taco Bell",
            "thumbnailUrl": "https://firebasestorage.googleapis.com/v0/b/weeat-1a169.appspot.com/o/
                profilePictures%2FMNGCmL1swnhpGgDMAx0GGQ8YUxy2%2FcreatedMenus%2F3g04nqjIqQVucYbUx8qa%2FTaco-Bell-Logo-1994.png?alt=media&token=c5806919-aedb-4f5b-9d40-3b6b83d90bd8"
        },
        {
            "restaurantName": "Red Orchid",
            "thumbnailUrl": "https://proxy-server-we-eat-e24e32c11d10.herokuapp.com/photo?
                photoreference=AelY_CsejQQmstd2tDHa4Bl172QrO59j7RuIQy4v7WbNAvjRW8yKWfkyLsU15rQ1QjvTAu4QxRNMdudNuEAK-kZ2dgJ7FuO1Z4gHA5NtGscs6I0jnBmBMqF3oyamcM5jNbs565SvipysZnLCy3XpYc4U1TSC_3MRUPnlyUu7hXK-C99WR-JE&maxwidth=400"
        },
        {
            "restaurantName": "Deniz Mediterranean Cuisine",
            "thumbnailUrl": "https://firebasestorage.googleapis.com/v0/b/weeat-1a169.appspot.com/o/
                profilePictures%2FprofilePictures%2FMNGCmL1swnhpGgDMAx0GGQ8YUxy2%2FcreatedMenus%2FDeniz%20Mediterranean%20Cuisine%2Fvanilla.jpg?alt=media&
                token=481ec319-b6a4-4120-a558-87b74f88ce52"
        },
        {
            "restaurantName": "El Sabor",
            "thumbnailUrl": "https://proxy-server-we-eat-e24e32c11d10.herokuapp.com/photo?
                photoreference=AelY_Cu9-haTCpd66155vsGuXYZQbP3W9YrHhvIaOaOz21uR_tFftI9RdRulhQHs7rL5O06TqBH2wq0Q3v8L5Pl2bfziceW0h26J1zaBozBXVJRNpYTfGimbJKk5dy_AgsLahw82bzxa0bQvQ-Z6BueWF359AS7zNI-hT9unDe4_HsrwxfuY&maxwidth=400"
        }
    ],
    "savedMenus": [
        {
            "restaurantName": "Subway"
        },
        {
            "restaurantName": "McDonald's"
        }
    ]
}
```


Error Response:

```
Forbidden
```

Diagram:



REVISION

- These results will now return the photo of the restaurants that the user created menus for. When the user creates the menu, the photo is stored in a ‘thumbnailUrl’, where, when displayed on the ‘Personalized Menu’ page, will show the image.
- For saved menus, the logo that is stored in the ‘restaurants’ database will render.

Get Saved Menus

- **Method: GET**
- **Endpoint: /users/{{userId}}/saved/{{menuId}}**
- **Purpose:** This endpoint is called when the user navigates to their personalized menu page. On that page, it renders a preview of the user's saved menu, with a button to view the menu, where the user can edit items and delete items.

Example Request:

```
GET    http://localhost:3001/users/{{userId}}/saved/{{menuId}}
```

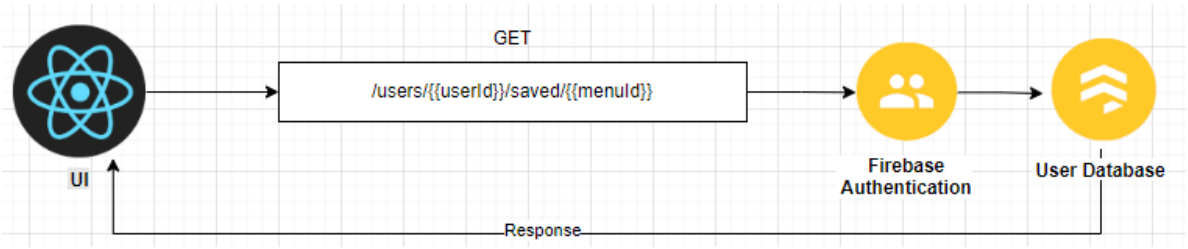
Success Response:


```
{
    "restaurantName": "McDonald's",
    "dishes": [
        {
            "note": "",
            "name": "Filet-O-Fish",
            "description": "Fish fillet sandwich with tartar sauce and cheese",
            "id": "cqJIFBjUSFaQgMdQBRvw",
            "category": "Fish",
            "allergens": [
                "Wheat",
                "Dairy",
                "Soy",
                "Fish"
            ]
        },
        {
            "name": "Chicken Nuggets",
            "description": "tender juicy",
            "category": "Chicken",
            "allergens": [
                "Wheat",
                "Soy"
            ],
            "id": "",
            "note": "I have a note now"
        },
        {
```


Error Response

```
Forbidden
```

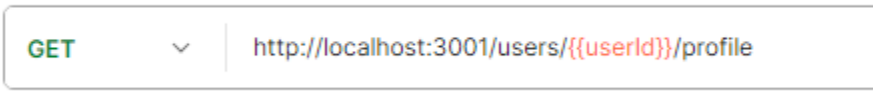
Diagram



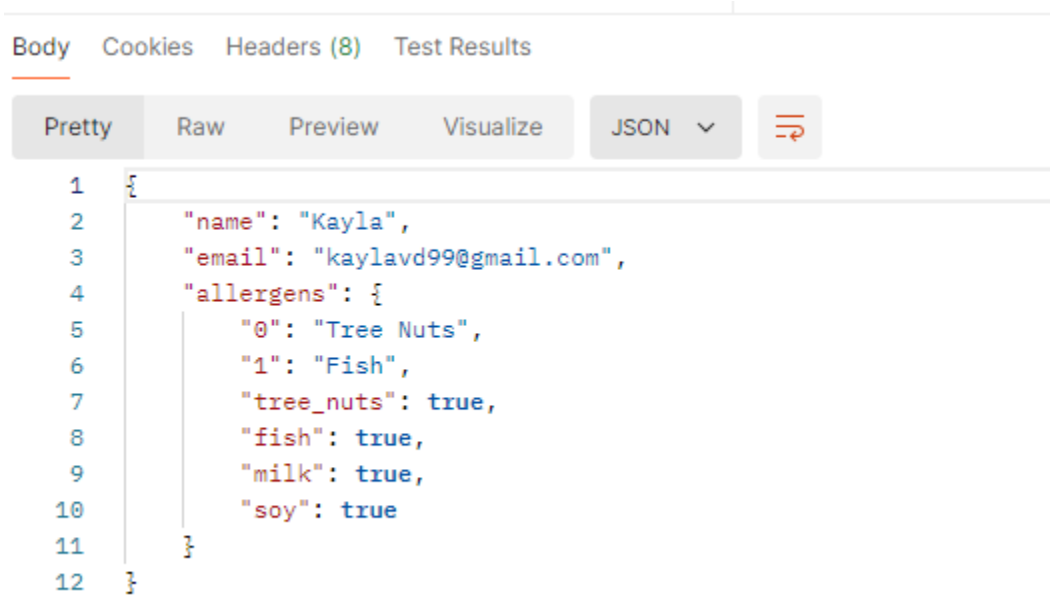
Get User Profile

- **Method: GET**
- **Endpoint: /users/{{userId}}/profile**
- **Purpose:** When the user logs into the application and navigates to their profile, the UI renders their profile information.

Example Request:

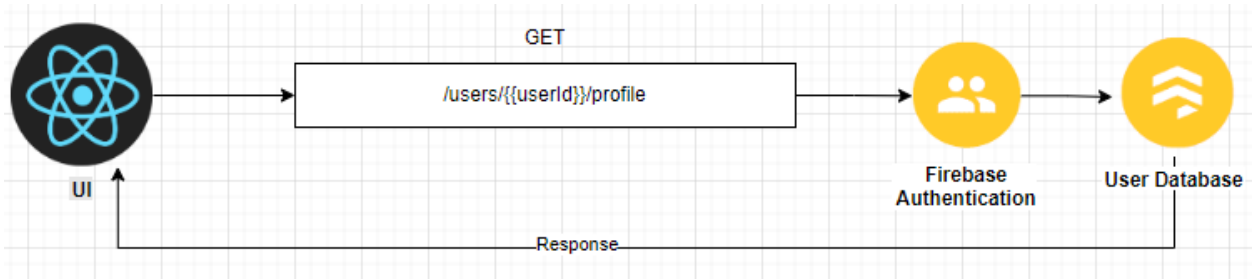


Success Response:



Error Response:

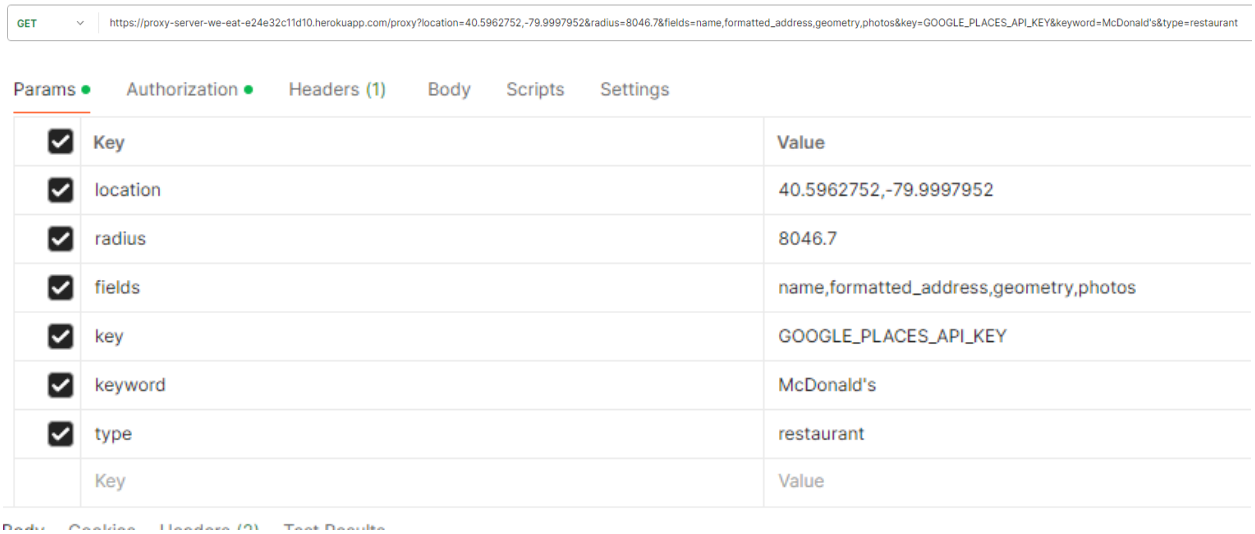
Diagram:



Search Restaurants - REVISED

- **Method: GET**
- **Endpoint:/textsearch?query={query}**
- **Purpose:** When a user navigates to the search page, Google's Geolocation API is called and returns the user's current coordinates. Then, this endpoint is called via the Heroku Proxy server to Google's Places API with the user's location and the restaurant name (keyword) to search for and restaurants within the defined radius are returned

Example Request:



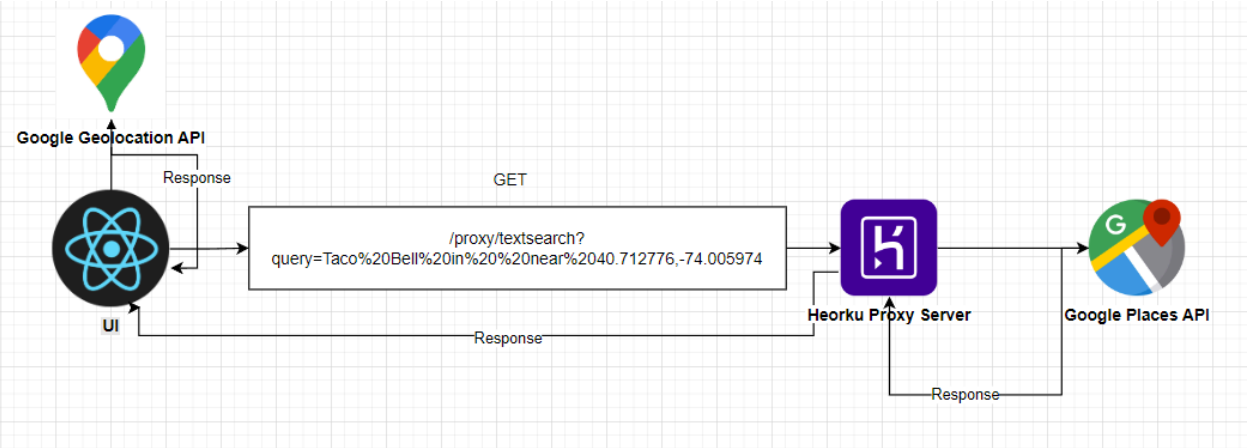
Success Response:


```
"results": [
    {
        "business_status": "OPERATIONAL",
        "formatted_address": "10955 Perry Hwy, Wexford, PA 15090, United States",
        "geometry": {
            "location": {
                "lat": 40.6175046,
                "lng": -80.0561413
            },
            "viewport": {
                "northeast": {
                    "lat": 40.61894057989272,
                    "lng": -80.05465327010728
                },
                "southwest": {
                    "lat": 40.61624092010727,
                    "lng": -80.05735292989273
                }
            }
        },
        "icon": "https://maps.gstatic.com/mapfiles/place_api/icons/v1/png_71/restaurant-71.png",
        "icon_background_color": "#FF9E67",
```

Error Response:

```
{
    "html_attributions": [],
    "results": [],
    "status": "ZERO_RESULTS"
}
```

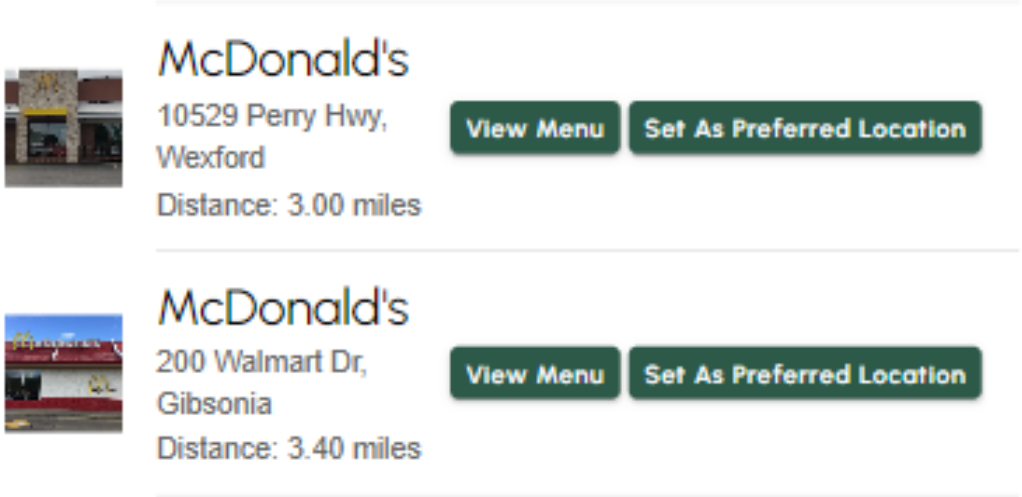
Diagram:



REVISION

- Previously, I did not pull in the restaurant's images from the Google Places API, but decided to add the location's images to each of the search results.
- Now, when the user clicks 'Search', the banner image of that restaurant's Google Business listing is fetched and displayed:

Example UI:



- The Proxy server will return a reference URL that use's the photo_reference attribute returned from the request:

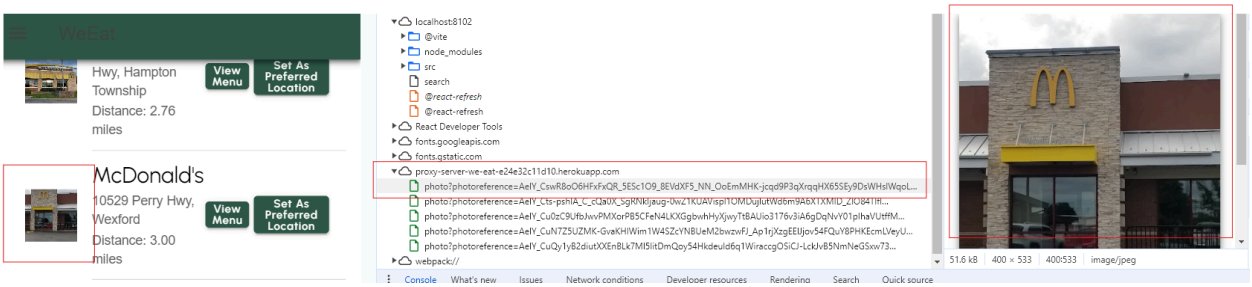
```
],
"photo_reference": "AelY_CuJhB1iURwd-Vjlg
```

JavaScript + Console:

```
const photoReference = result.photos?.[0]?.photo_reference;
console.log("Photo reference: ", photoReference)
```

```
Photo reference:  AelY_CswR8oO6HFxFxQR_5ESc1O9_8EVdXF5_NN_OoEmMHK-jcqd9P3qXrqqHX65SEy9DsWHs1WqoLBNfjAfdoJEzGRQSb-O_Z-sA3bImWdovDN6yxBHGP-Jm5-JTccmw73nPgoC47rzHYqeFd2Gzmvgk4twvNzqRNfXt2UhWvzUevbbj9Ca                searchService.ts:83

Photo url:                                                                                                                                    searchService.ts:85
https://proxy-server-we-eat-e24e32c11d10.herokuapp.com/photo?photoreference…Jm5-JTccmw73nPgoC47rzHYqeFd2Gzmvgk4twvNzqRNfXt2UhWvzUevbbj9Ca&maxwidth=400
```



Get Restaurant's Full Menu

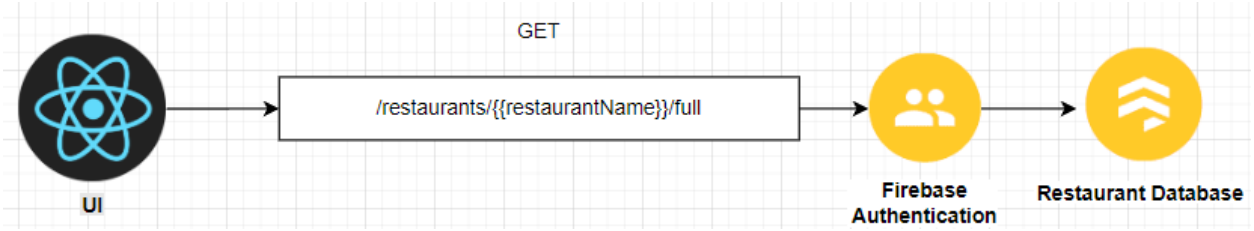
- **Method: GET**
- **Endpoint: /restaurants/{{restaurantName}}/full**
- **Purpose:** When the user goes to view their saved menu from a menu already in the restaurant database (i.e., ones 'offered' on the application), they have the option to go to the restaurant's full menu. When clicking the 'View Full Menu', the endpoint is called and the all menu items from that restaurant are rendered. This is also used in the Search page when the user clicks on a search result for a menu that is in the restaurant database. If the restaurant's menu is not within the user's savedMenus, the full menu is rendered.

Example Response:

Successful Response

Error Response:

Diagram:



Create a Menu

- **Method: POST**
- **Endpoint:/users/{{userId}}/menus/created/{{menuId}}/dishes**
- **Purpose:** When a user navigates to the 'Create Menu' page from their profile, they input the relevant information and when the 'Create' button is clicked, this endpoint is called and inserts the menu into the user's 'Created Menus' subcollection.

Example Request:

POST http://localhost:3001/users/{{userId}}/menus/created

```
{
  "menu": {
    "restaurantName": "Example Restaurant",
    "dishes": [
      {
        "name": "Dish 1",
        "description": "Delicious dish",
        "allergens": ["Peanuts"],
        "note": "Spicy",
        "category": "Main Course"
      }
    ]
  }
}
```

Success Response:

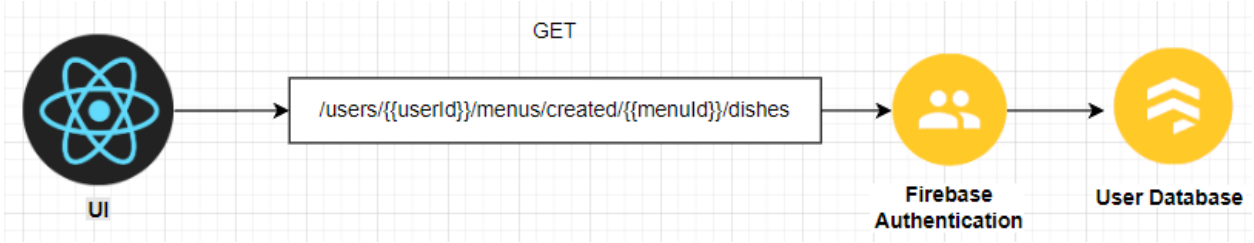
Pretty Raw Preview Visualize HTML

```
Menu item added successfully
```

Error Response

```
Error adding menu item: Value for argument "data" is not a valid Firestore document. Input is not a plain JavaScript
object.
```

Diagram:



Editing Allergies

- **Method: PUT**
- **Endpoint:/users/{{userId}}/allergens**
- **Purpose:** When a user navigates to their profile and clicks 'Edit Allergens', they are brought to a form where they can edit their allergens via checkboxes and their changes are reflected in their user profile.

Example Request:


```
{
  "allergens": {
    "Peanuts": true,
    "Shellfish": false,
    "Dairy": true
  }
}

```

Success Response:

```
User allergens updated successfully
```

Error Response:

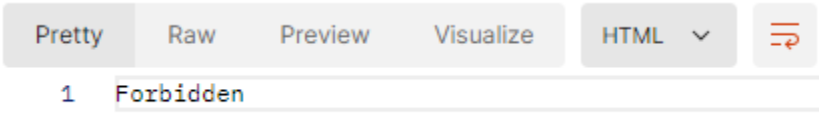
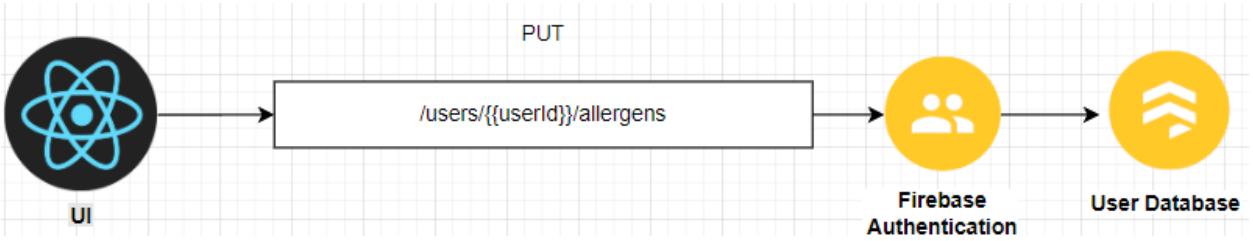


Diagram:

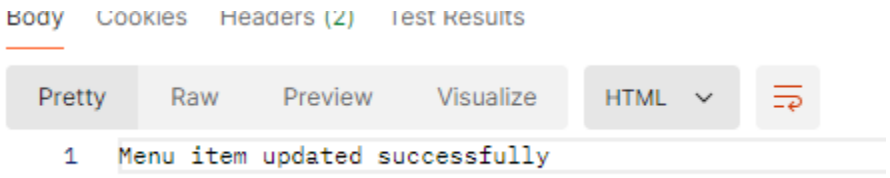


Editing Item in Created Menu

- **Method: PUT**
- **Endpoint:/users/{{userId}}/menus/created/{{menuId}}/dishes/{{createdMenuItemId}}**
- **Purpose:** When a user wants to edit a menu item they created, they can do so by going to their personzalied menus page where they see links to their Created Menus. From there, on their respective created menu pages, they can edit all the fields of the menu item.

Example Request:

Success Response:



Error Response:

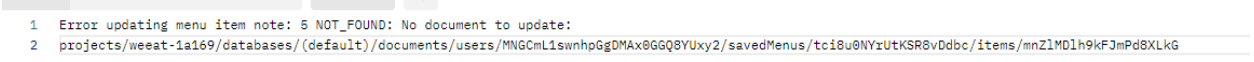
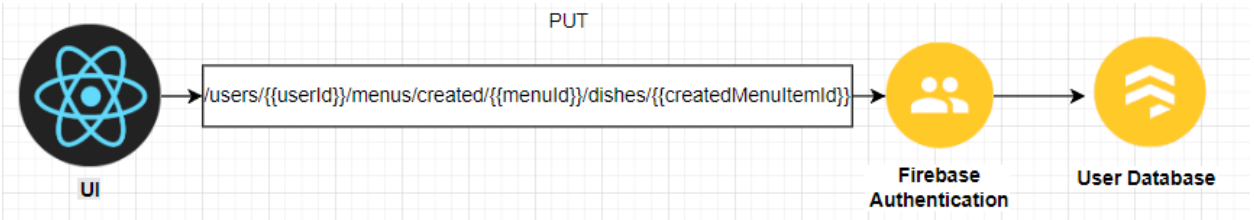


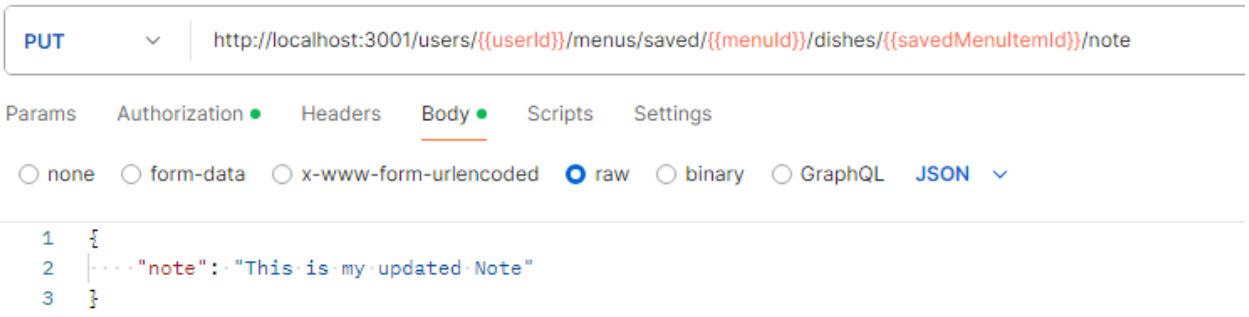
Diagram:



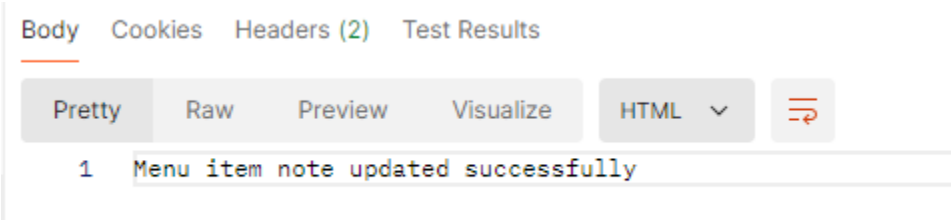
Editing Note in Saved Menu

- **Method: PUT**
- **Endpoint:/users/{{userId}}/menus/saved/{{menuId}}/dishes/{{savedMenuItemId}}/note**
- **Purpose:** When a user wants to edit a menu item they saved, they can only edit the notes of the menu item. When on their saved menu page for the respective menu, they click a, “Edit Item” button and modal pops up allowing them to edit the note.

Example Request:



Success Response:



Error Response:

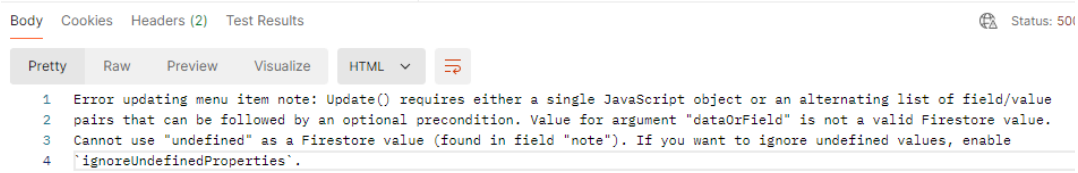
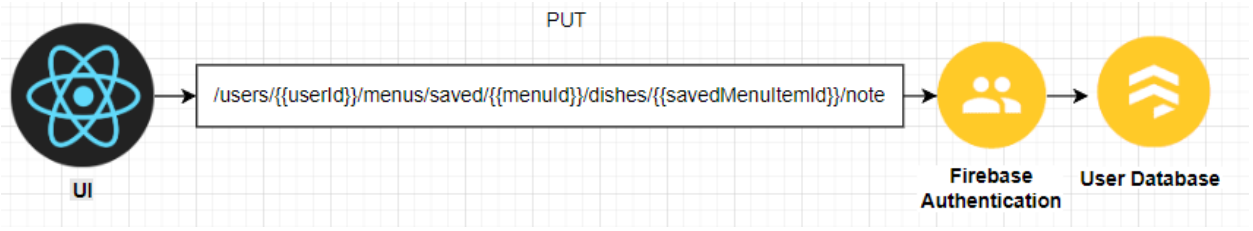


Diagram:

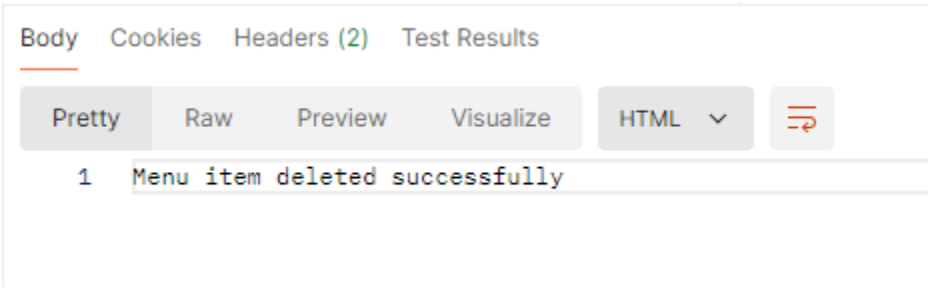


Delete Item from Saved Menu

- **Method: DELETE**
- **Endpoint:/users/{{userId}}/menus/saved/{{menuId}}/dishes/{{savedMenuItemId}}**
- **Purpose:** When a user wants to delete a menu item from their saved menus, they navigate to their respective saved menu and on that respective page, they click a delete button that confirms whether or not they want to delete the item and once confirmed, this endpoint is called.

Example Request:

Success Response:



Error Response:

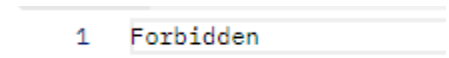
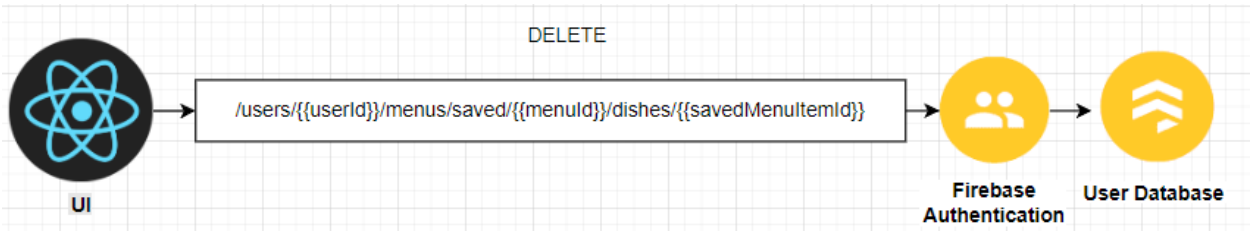


Diagram:



Deleting Item from Created Menu

- **Method: DELETE**
- **Endpoint:/users/{{userId}}/menus/created/{{menuId}}/dishes/{{createdMenuItemId}}**
- **Purpose:** When a user wants to delete a menu item from their created menus, they navigate to their respective created menu and on that respective page, they

click a delete button that confirms whether or not they want to delete the item and once confirmed, this endpoint is called.

Example Request:

```
DELETE  http://localhost:3001/users/{{userId}}/menus/created/{{createdMenuId}}/dishes/RTNM4YjAlXrnFzLEMeu9
```

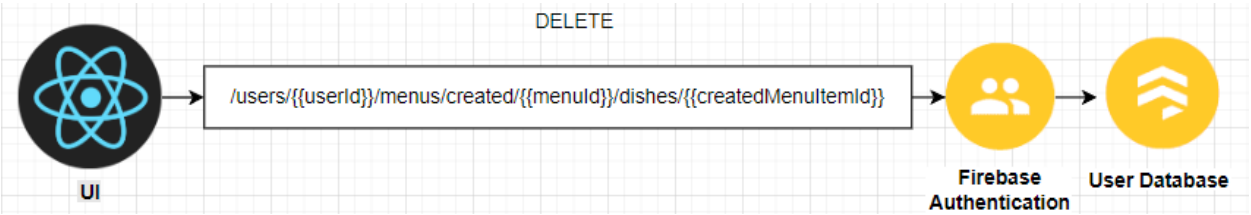
Success Response:

```
1  Menu item deleted successfully
```

Error Response:

```
1  Forbidden
```

Diagram:

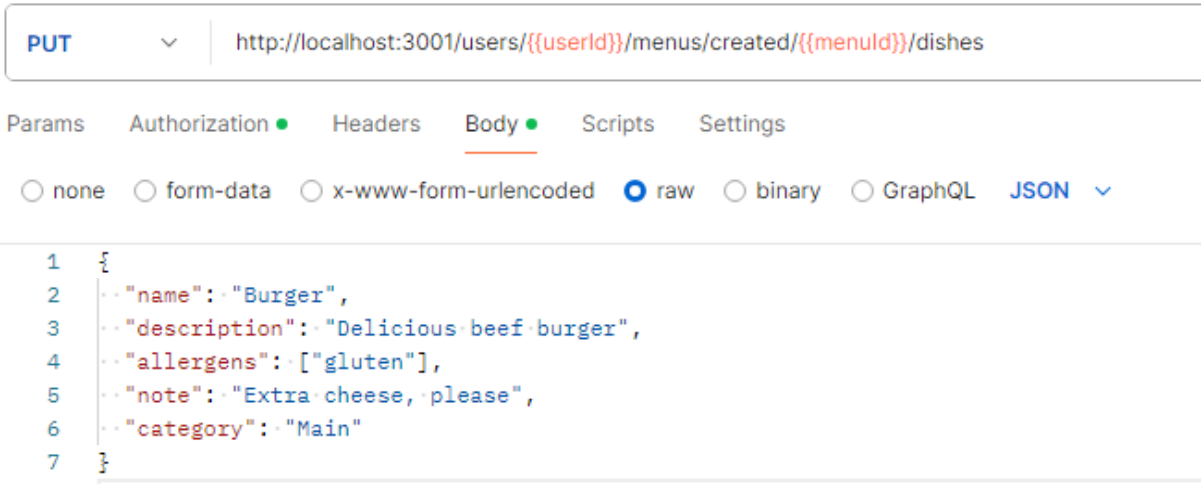


Adding Item to Created Menus

- **Method: PUT**
- **Endpoint:/users/{{userId}}/menus/created/{{menuId}}/dishes/**
- **Purpose:** When a user wants to add a menu item to their created menus, they navigate to the respective created menu's page, where there is an 'Add Menu

Item' button. Clicking this button invokes a modal to appear, where the user inputs the required parameters and that menu item is added to that specific created menu in their createdMenus subcollection.

Example Request:



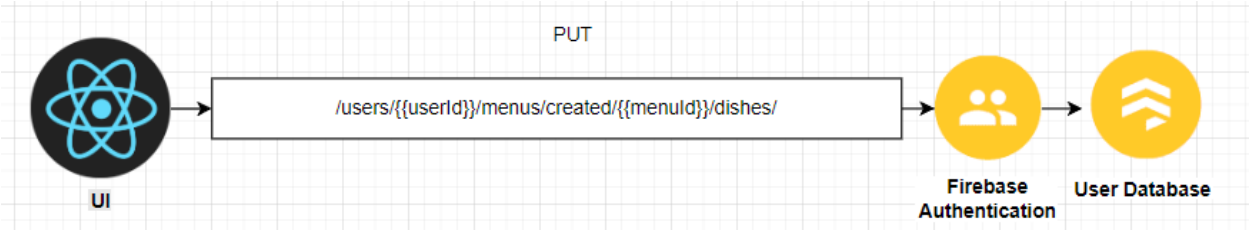
Success Response:

```
Menu item added successfully
```

Error Response:

```
Forbidden
```

Diagram:



Stretch Endpoints

Add a Review

- **Method: POST**

- **Endpoint:/users/{{userId}}/reviews**
- **Purpose:** When a user wants to leave a review for a restaurant, they will leave a review for that restaurant but categorized by the location, since you can switch preferred locations for the same restaurant. These reviews would be stored in a subcollection of a user document named, ‘reviews’

Example Request:

Params:

```
{
  "review": {
    "restaurantName": "Red Orchid",
    "locationAddress": "123 Main St",
    "starRating": 5,
    "review": "Great food!"
  }
}
```

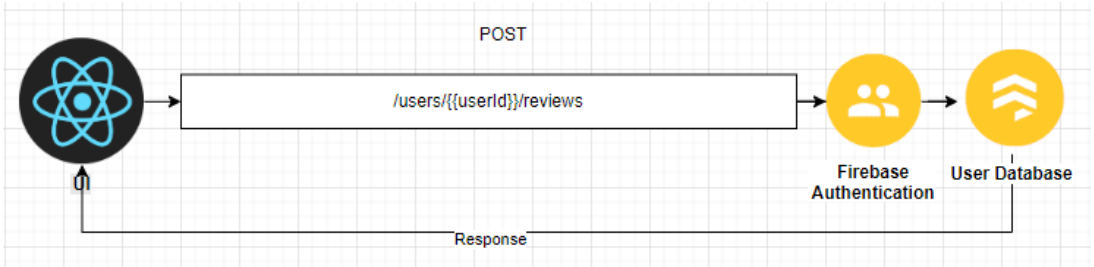
Success Response:

```
{
  "message": "Review added successfully"
}
```

Error Response

```
{
  "error": "Unauthorized",
  "message": "Invalid token"
}
```

Diagram:



Get User Reviews

- **Method: GET**
- **Endpoint:/users/{{userId}}/reviews**

- **Purpose:** Fetch all of the reviews that the user left

Example Request:

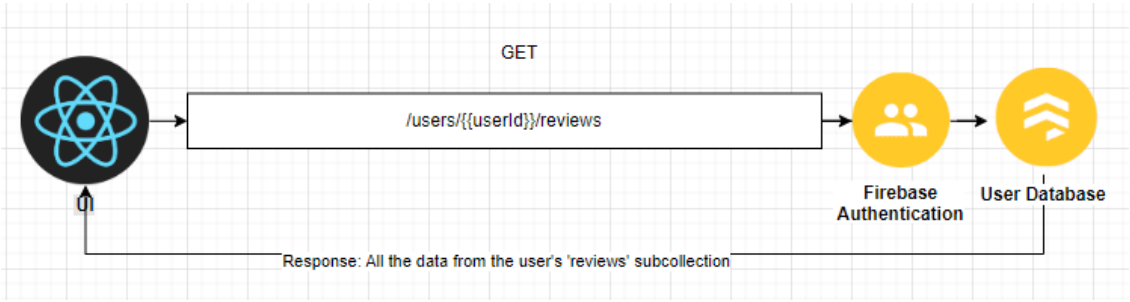
Success Response:

```
[
  {
    "restaurantName": "Red Orchid",
    "locationAddress": "123 Main St",
    "starRating": 5,
    "review": "Great food!"
  },
  {
    "restaurantName": "Green Bamboo",
    "locationAddress": "456 Elm St",
    "starRating": 4,
    "review": "Nice ambiance."
  }
]
```

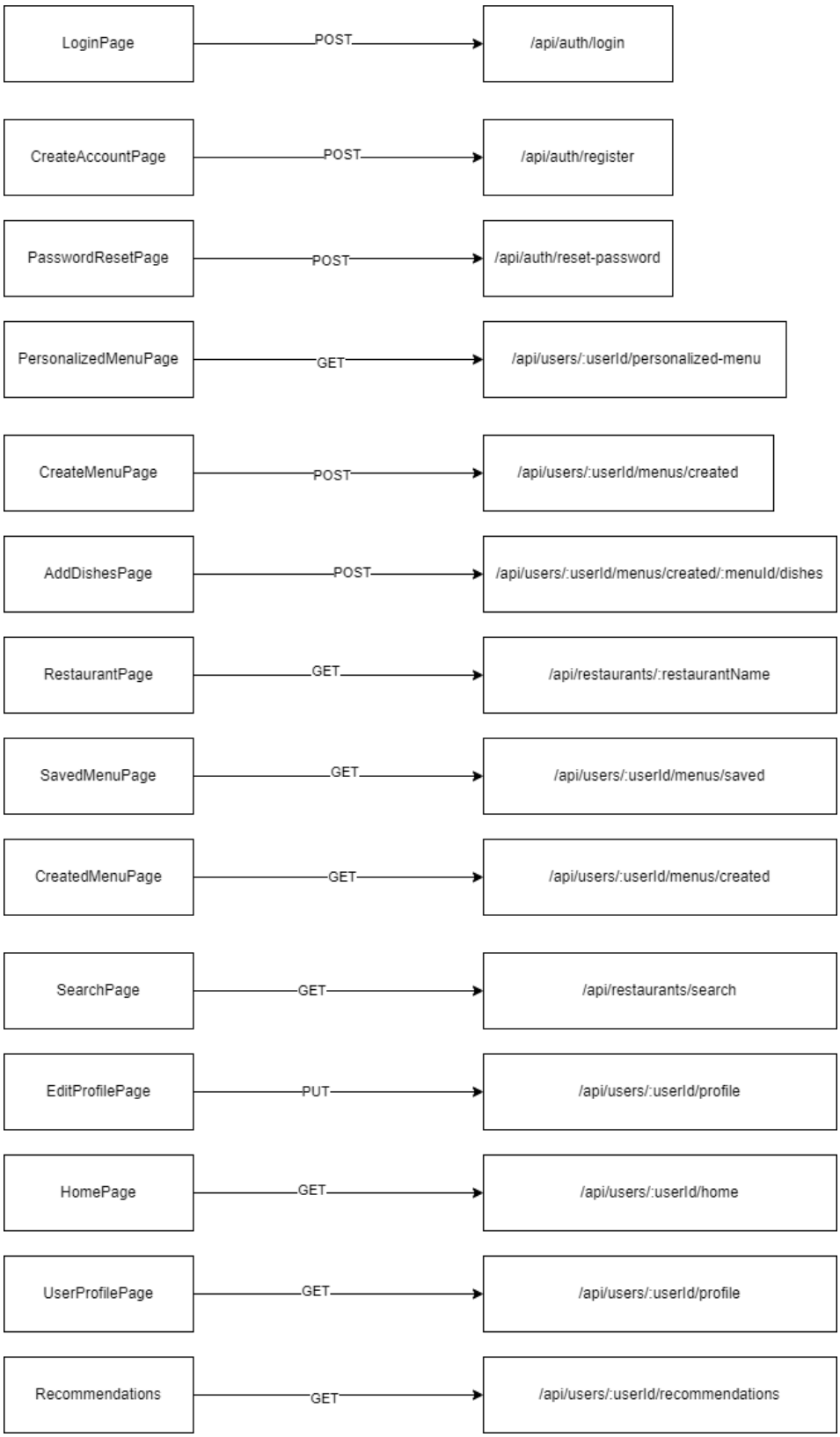
Error Response

```
{
  "error": "Unauthorized",
  "message": "Invalid token"
}
```

Diagram:



Communication Diagram: MVP



Stretch Features (in addition to the MVP)