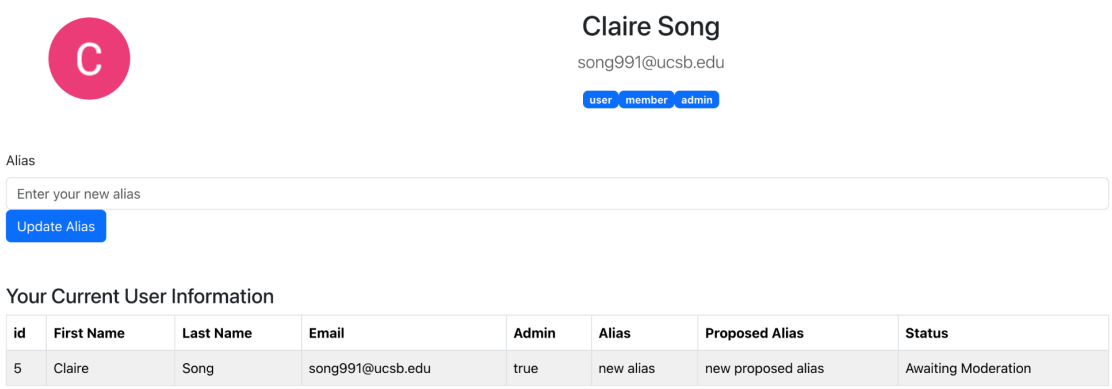
Release Notes

proj-dining, f24-16

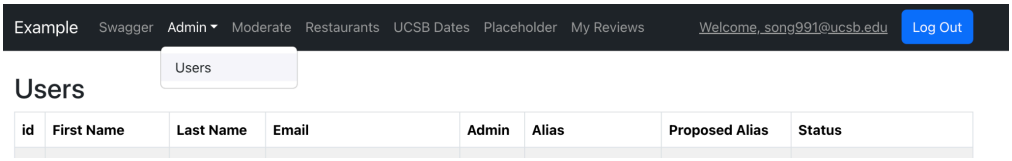
Features for the Users:

1. Profile page now displays user's information and an input field to enter a new alias.
    - Users can enter a new alias, which will then be reflected on the user information table under the Proposed Alias column with the status Awaiting Moderation. The default alias is Anonymous User.

    - After user proposes a new alias:
        - If an admin approves it, the proposed alias will become the Alias, and the status will update to "Approved on [date approved]"
        - If rejected, the alias would stay the same and the status will update to "rejected."



    - The Users table now shows the columns Alias, Proposed Alias, and Status

- 

2. Dining Commons table has been implemented.
    - A Dining Commons table is now viewable, with columns that list the dining common's code, name, location, and qualities pertaining to whether or not a dining cam, takeout meal, or sack meal exists.

    - No page has been implemented at this time. Later, it will be developed so that users will see the table on the home page directly.

Dining Commons Table *Empty, User view, Admin view*

Empty

Code	Name	Has Dining Cam	Has Sack Meal	Has Takeout Meal	Latitude	Longitude

Show code

Three Items Ordinary User

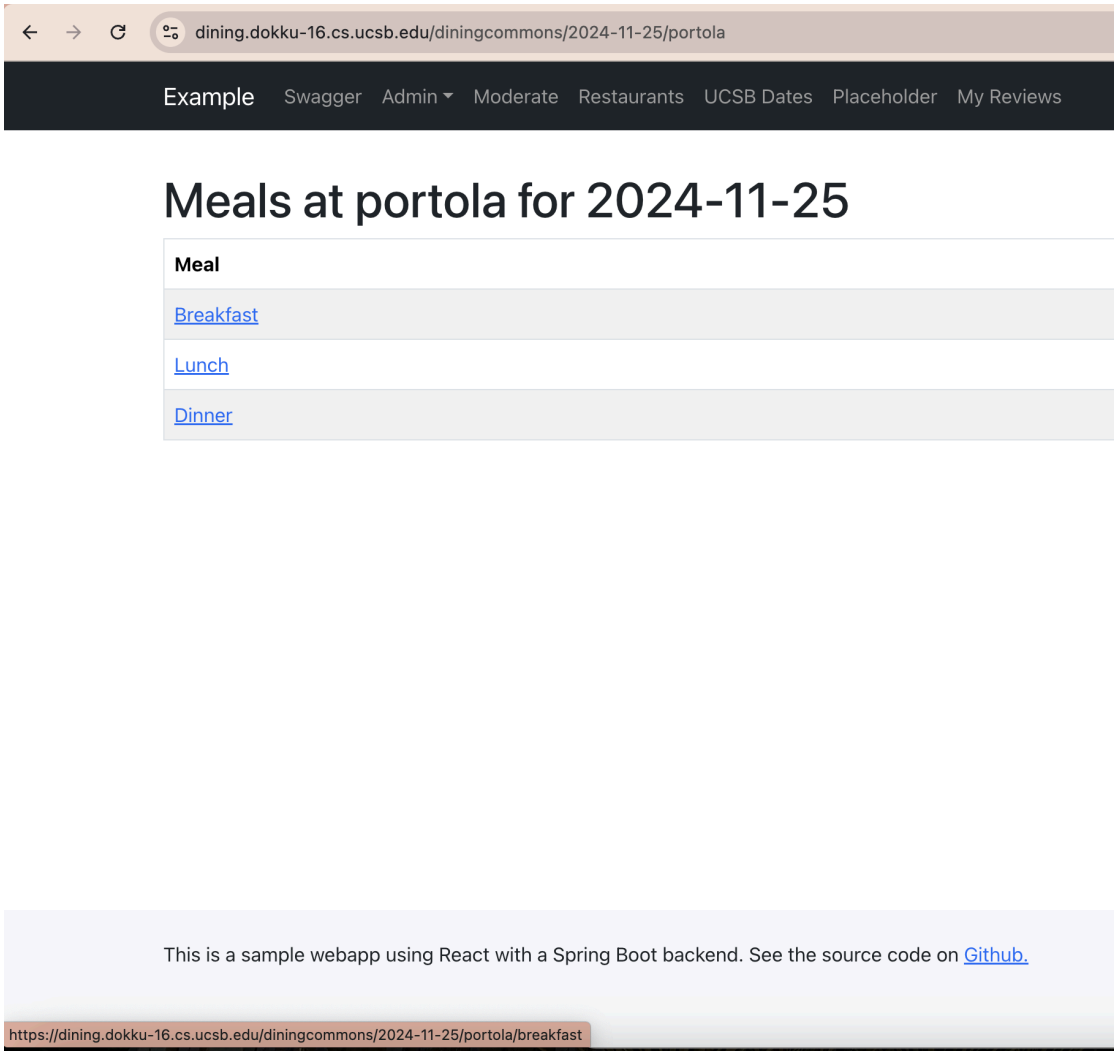
Code	Name	Has Dining Cam	Has Sack Meal	Has Takeout Meal	Latitude	Longitude
carrillo	Carrillo	✅	❌	❌	34.409953	-119.85277
de-la-guerra	De La Guerra	✅	❌	❌	34.409811	-119.845026
ortega	Ortega	✅	✅	✅	34.410987	-119.84709
portola	Portola	✅	✅	✅	34.417723	-119.867427

Show code

Three Items Admin User

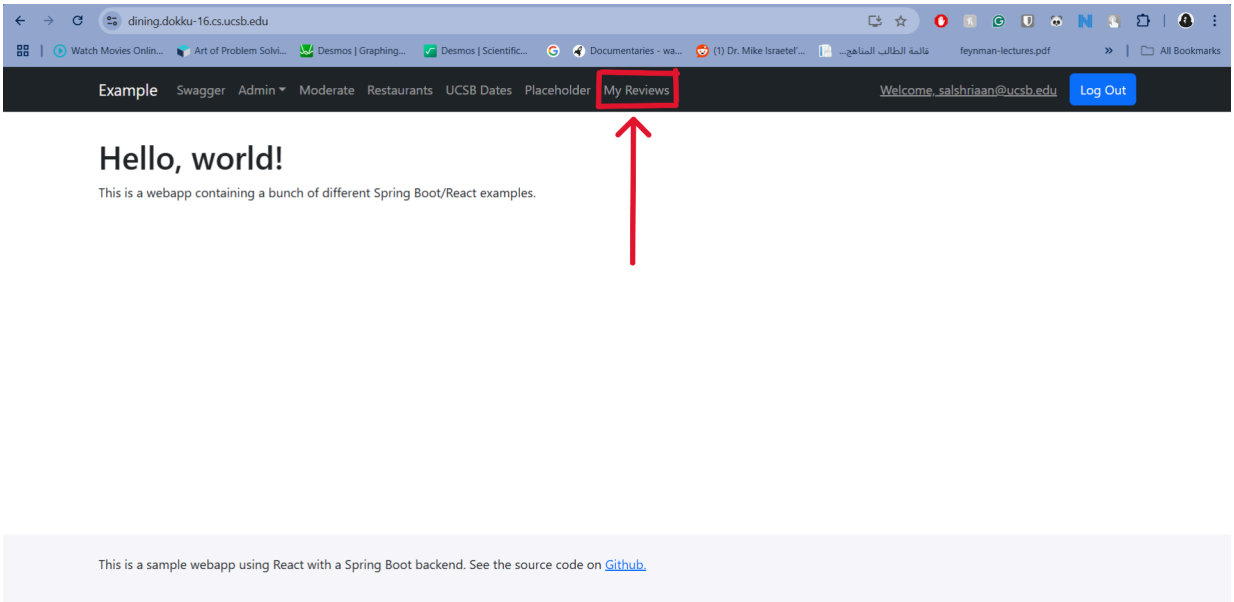
Code	Name	Has Dining Cam	Has Sack Meal	Has Takeout Meal	Latitude	Longitude
carrillo	Carrillo	✅	❌	❌	34.409953	-119.85277
de-la-guerra	De La Guerra	✅	❌	❌	34.409811	-119.845026
ortega	Ortega	✅	✅	✅	34.410987	-119.84709
portola	Portola	✅	✅	✅	34.417723	-119.867427

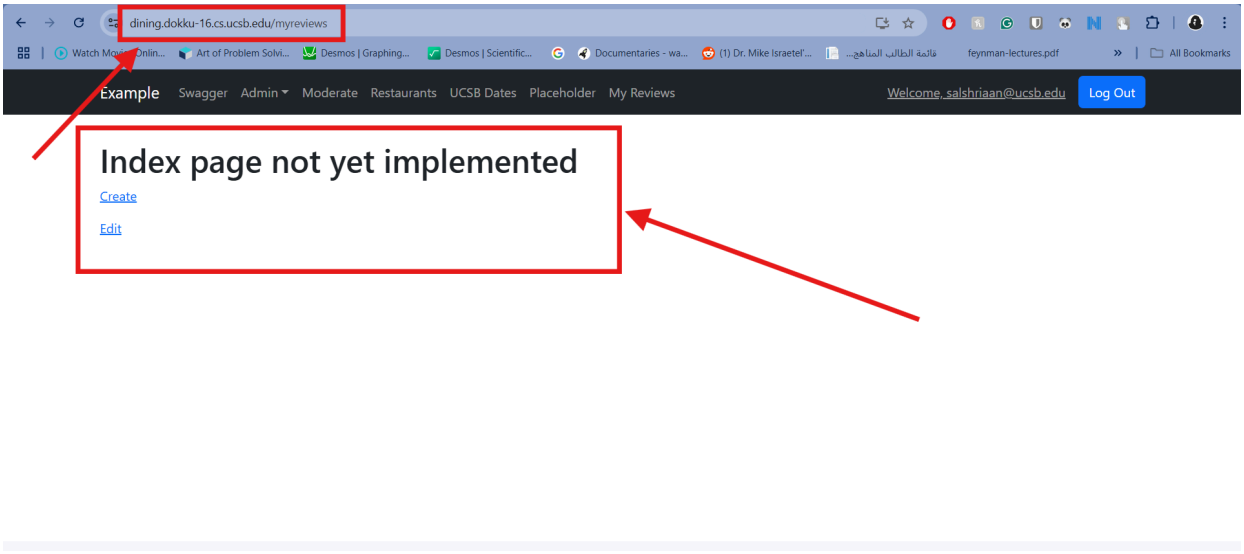
3. Users can now navigate to a page based on the date and the dining common using the search bar at /diningcommons/:date-time/:dining-commons-code to see the meal times that that dining hall is serving.
    - Every meal time on the table links to a page /diningcommons/:date-time/:dining-commons-code/meal-code which shows the menu items in a table that hasn't been implemented yet.



4. A MyReviews button now appears on the navigation bar.
    - When clicked, this button takes the user to the url : (https://dining.dokku-16.cs.ucsb.edu/myreviews)

    - This is a placeholder page for now, but in the future, it is supposed to show and list all the individual reviews that the user has made on there, hence why it is called “MyReviews”.





Summary of Implementations for Developers:

1. Dining commons controller.
   - Added a controller that uses a get type endpoint and maps such a request with the pathing "/all".

   - Utilizes the Dining Commons service and the .get() handle to obtain all Dining Commons present from the UCSB Developer API.
2. Dining commons service with .get() endpoint.
   - Added a service API to obtain fields from the UCSB Developer Dining Commons API's /get.

   - Utilized by the Dining Commons controller.

Prior to Execution

Post-execution

API to handle get all dining commons from UC SB developer API website

GET /api/dining/all Get all Dining Commons

Parameters

Cancel

No parameters

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'https://dining.dokku-16.cs.ucsb.edu/api/dining/all' \
  -H 'accept: application/json' \
  -H 'X-XSRF-TOKEN: 7910731e9-1b38-4ef7-ad56-697831c907aa'
```

Request URL

https://dining.dokku-16.cs.ucsb.edu/api/dining/all

Server response

Code	Details
200	Response body


```
[
  {
    "name": "Carrillo",
    "code": "carrillo",
    "hasDiningCam": true,
    "hasSackMeal": false,
    "hasTakeOutMeal": false,
    "latitude": null,
    "longitude": null
  },
  {
    "name": "De La Guerra",
    "code": "de-la-guerra",
    "hasDiningCam": true,
    "hasSackMeal": false,
    "hasTakeOutMeal": false,
    "latitude": null,
    "longitude": null
  },
  {
    "name": "Ortega",
    "code": "ortega",
    "hasDiningCam": true,
    "hasSackMeal": true,
    "hasTakeOutMeal": true,
    "latitude": null,
    "longitude": null
  },
  {
    "name": "Portola"
```


Download

Response headers

```
cache-control: no-cache,no-store,max-age=0,must-revalidate
content-encoding: gzip
content-type: application/json
date: Thu,05 Dec 2024 00:25:10 GMT
expires: 0
pragma: no-cache
server: nginx
strict-transport-security: max-age=31536000 ; includeSubDomains
vary: Accept-Encoding
x-content-type-options: nosniff
x-firefox-spdy: h2
x-frame-options: DENY
x-xss-protection: 0
```

Responses

Code	Description	Links
200	OK	No links
404	Not Found	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

```
{}
```

Media type: */*

Example Value | Schema

```
{}
```

3. Dining Commons Menu Controller

- Added API endpoint to get the meal times based on the given date time and dining common.

- This was used for the Meal Times table at /diningcommons/:date-time/:dining-commons-code.

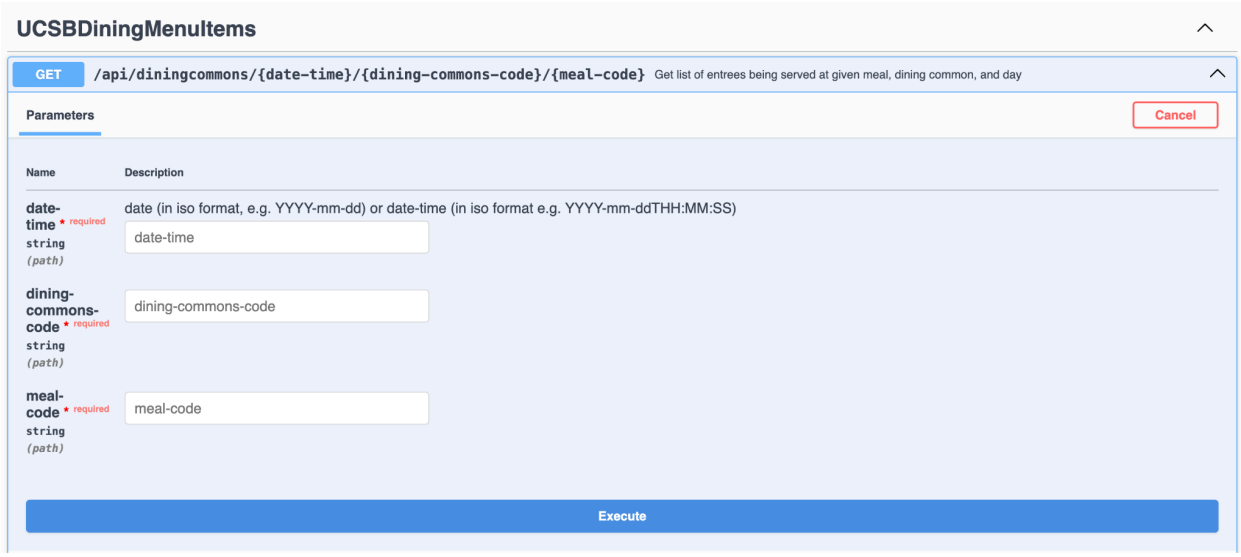
- For future developers, the page at /diningcommons/:date-time/:dining-commons-code should include a dropdown box for the users to select a date, so users can view meal times at different dates easily.

4. Dining Commons Menu Items Controller

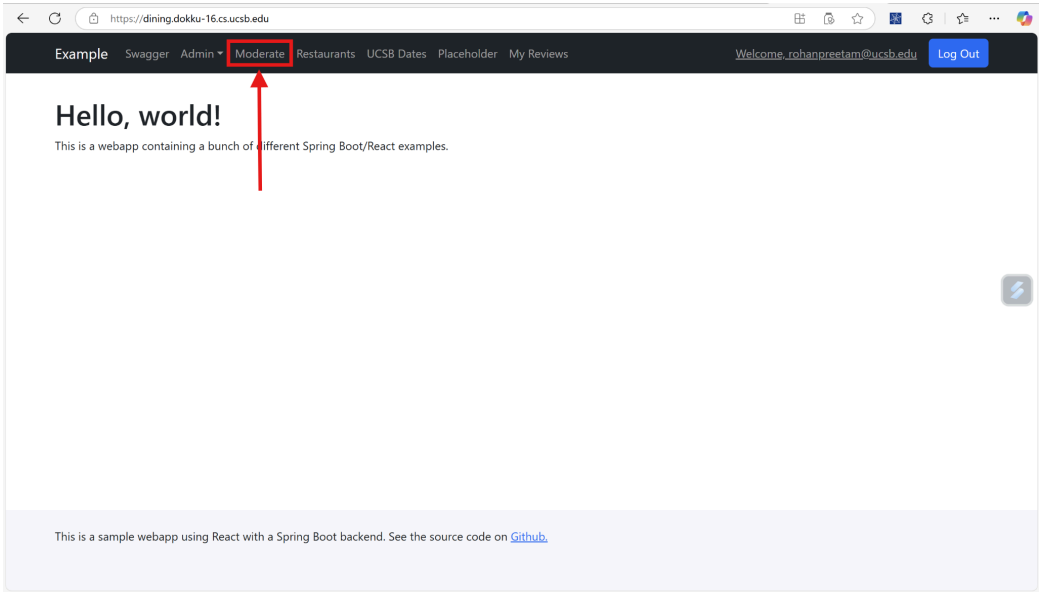
- Added the API endpoint to get the menu items given a date time, dining common, and meal time.

- This returns a list of menu items and its station with the dining common and meal code. Each menu item includes an id that stays in the database, so that if that same menu item (with same dining common and meal code) appears again, it would return the first id that was set along with the first instance of that menu item.

- This has not yet been integrated into the app, but this can be used in making the Menu Items Table as well as for the Reviews Table.



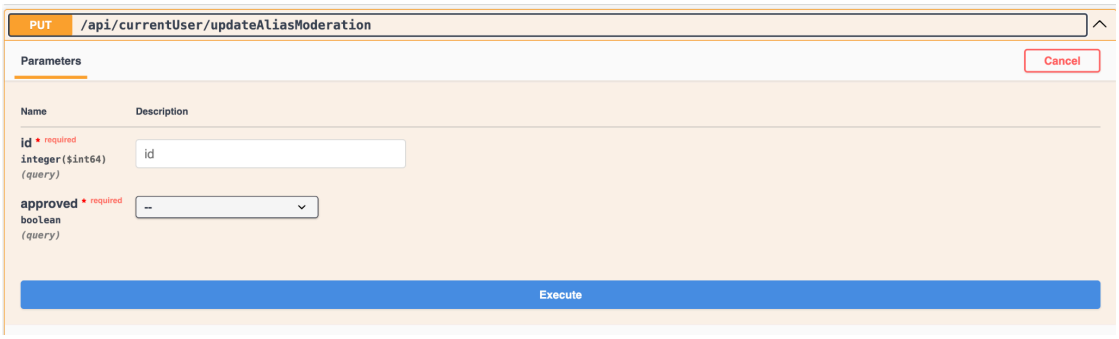
5. Moderate button on the navigation bar (appears only for admin users):
   - There is only a placeholder for now. It takes the user to a page only if they have admin access.



6. User information controller
   - Added post endpoint for users to propose a new alias

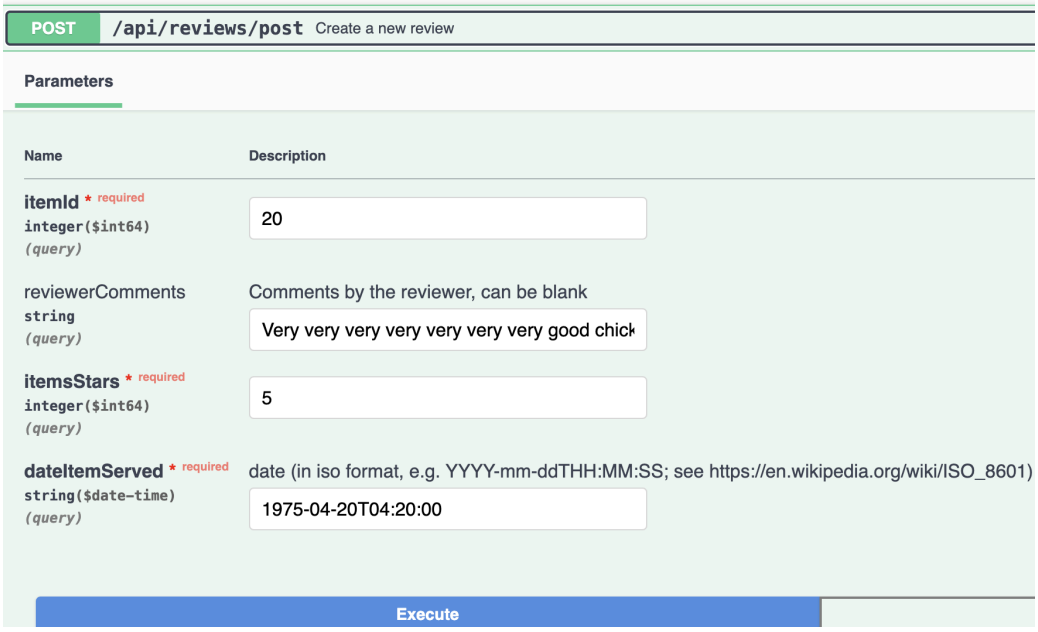
- Added get endpoint for admin to retrieve a list of all users who proposed a new alias.

- Added put endpoint for admin to enter a user id and approve/reject the user's proposed alias.



7. Reviews Feature:

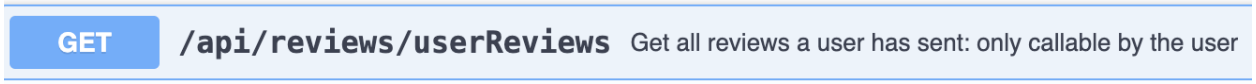
- Currently, there exists an API that allows users to create reviews and get all the reviews that they have created and submitted. On creation of a review, the user <u>must</u> supply three pieces of data: the item ID for the meal, the number of stars (a rating 1- 5), and the date the item was served. Additionally, the user may add text-based data, aka the review aspect, but this is optional.



- Other pieces of data are not exposed to the user but are initialized on the creation of a review. These pieces are the time that the review was created, the time that the review was edited (initialized to the creation time), the status of the review (initialized to “Awaiting Moderation”), the ID of the user, and data about the moderator (Moderators ID, Moderators comments) initialized to null.

- There are some pre-requirements before a user can successfully add a review, specifically, the database that holds all the food items and associated data must be populated with data. Currently, this is manually done but must be done before a review can be submitted or else it will fail. The best place for this to programmatically happen is through a frontend feature that looks up all the items that were served on a specific day, meal time, and dining hall. This would inherently populate the database with valid data and allow the user to visually choose which item they had, thus abstracting the need to manually choose or know the item ID. However, once this is done, the review will successfully be submitted as long as the item is in the food database, else it will return a 404 error.

- The Swagger interface for getting all of the users reviews (user callable method to return users data only) needs no parameters. It returns a list every review a user has submitted and has the full scope of data stated above.



- The return is structured as follows:


```
[
  {
    "id": 1,
    "studentId": 8,
    "itemId": 20,
    "reviewerComments": "Very very very very very very very good chicken.",
    "itemsStars": 5,
    "dateItemServed": "1975-04-20T04:20:00",
    "status": "Awaiting Moderation",
    "userIdModerator": null,
    "moderatorComments": null,
    "dateCreated": "2024-12-05T00:46:55.340126",
    "dateEdited": "2024-12-05T00:46:55.340126"
  },
  {
    "id": 2,
    "studentId": 8,
    "itemId": 3,
    "reviewerComments": "Very upset by the food today, found metal shavings in it.",
    "itemsStars": 1,
    "dateItemServed": "1776-07-13T12:12:12",
    "status": "Awaiting Moderation",
    "userIdModerator": null,
    "moderatorComments": null,
    "dateCreated": "2024-12-05T03:45:24.947829",
    "dateEdited": "2024-12-05T03:45:24.947829"
  }
]
```


- The other integrated API endpoint allows for an admin to receive all reviews: that is reviews that have been rejected, accepted, or are still in the process of being reviewed. These other states of a review are still in development, so all reviews will be labeled with “Awaiting Review”. This method is only callable by an admin and returns a list structured in the same format as above.