

# User Changes

- There is a new menu “My Reviews” which shows all reviews submitted by the user

This can be found at the address /myreviews. It displays all the reviews of the current user. Notice how the ReviewID is the same, indicating that it’s only the reviews from that specific user.

Currently does not have the Edit or Delete buttons enabled.

Example

Swagger

Admin ▾

Moderate

Restaurants

UCSB Dates

Placeholder

My Reviews

Welcome, emilyzheng@ucsb.edu

Log Out

My Reviews

Create Reviews

id	Reviewer ID	Item ID	Date Served	Stars	Review Text	Status	Mod ID	Moderator Comments	Created Date	Last Edited Date
1	1	1	2023-11-01T16:00:00	3	yummy	Awaiting Moderation			2024-11-25T09:24:32	2024-11-25T09:24:32
3	1	1	2023-11-01T16:00:00	2	good food	Awaiting Moderation			2024-11-28T02:31:26	2024-11-28T02:31:26

- Added new placeholder pages for features that will eventually be filled with relevant information.

First there is a placeholder at address /diningcommons/:diningcommonscode, which will change its placeholder text depending on the url field /:diningcommonscode.

Here is what /diningcommons/:carillo would look like:

Example	H2Console	Swagger	Admin ▾	Restaurants	UCSB Dates	Placeholder	Welcome, stevenjiang@ucsb.edu	Log Out
---------	-----------	---------	---------	-------------	------------	-------------	-------------------------------	---------

## Placeholder for Dining Commons Page for carillo

Second, there is a placeholder at address /reviews/:itemid.

This page will eventually show reviews for an item

Lastly, there is a placeholder at address /moderate, which can be clicked from the Navbar button labeled “Moderate”.

This page will eventually show the moderation controls

- There is a DiningMenuApiController accessible on Swagger that returns relevant information from the API

DiningMenuApiController	
GET	/api/dining/getMeals Get list of meals served in given dining commons on given date
GET	/api/dining/getDays Get list of days with meal service
GET	/api/dining/getCommons Get list of dining commons serving meals on given date

- The first endpoint, getDays(), returns the list of days meal services are being offered at the dining commons, based on data from the API

Code	Details
200	<div>Response body</div> <pre>[   {     "name": "Today",     "code": "2024-12-04",     "date": "2024-12-04T00:00:00-08:00"   },   {     "name": "Tomorrow",     "code": "2024-12-05",     "date": "2024-12-05T00:00:00-08:00"   },   {     "name": "Friday",     "code": "2024-12-06",     "date": "2024-12-06T00:00:00-08:00"   },   {     "name": "Saturday",     "code": "2024-12-07",     "date": "2024-12-07T00:00:00-08:00"   }, ]</pre>

- The second endpoint, `getCommons()`, takes in a day and returns the dining commons that are serving meals during that time

GET

/api/dining/getCommons

Get list of dining commons serving meals on given date

^

Parameters

Cancel

Name	Description
<b>dateTime</b> • required string(\$date-time) (query)	<input type="text" value="2024-12-04T00:00:00-08:00"/>

Execute

Code

Details

200

Response body

```
[
  {
    "name": "Carrillo",
    "code": "carrillo"
  },
  {
    "name": "De La Guerra",
    "code": "de-la-guerra"
  },
  {
    "name": "Ortega",
    "code": "ortega"
  },
  {
    "name": "Portola",
    "code": "portola"
  }
]
```

- The last endpoint, `getMeals()`, takes in a day and dining commons, to return which meals are being served at the given time and place

GET

/api/dining/getMeals

Get list of meals served in given dining commons on given date

^

Parameters

Cancel

Name	Description
<b>dateTime</b> • required string(\$date-time) (query)	<input type="text" value="2024-12-04T00:00:00-08:00"/>
<b>diningCommonsCode</b> • required string (query)	<input type="text" value="carrillo"/>

Execute

Code

Details

200

Response body

```
[
  {
    "name": "Breakfast",
    "code": "breakfast"
  },
  {
    "name": "Lunch",
    "code": "lunch"
  },
  {
    "name": "Dinner",
    "code": "dinner"
  }
]
```

- In addition, if an invalid date and time is entered, an error message is returned by the `getCommons()` and `getMeals()` endpoint

## Dev Changes

### 1. New database columns added to “users” database

Added the following rows to the users table:

- `alias` (String) - The current alias of the user
- `status` (String) - The state of the alias. Can be “Approved”, “Pending”, or null
- `alias_pending` (String) - When the user enters an alias, it will be stored in this field until a moderator approves it
- `date_approved` - The date the current alias was approved

This is what the fields of the new users table look like:

Column	Type	Collation	Nullable	Default
<code>id</code>	<code>bigint</code>		<code>not null</code>	<code>generated by default as identity</code>
<code>admin</code>	<code>boolean</code>		<code>not null</code>	
<code>email</code>	<code>character varying(255)</code>			
<code>email_verified</code>	<code>boolean</code>		<code>not null</code>	
<code>family_name</code>	<code>character varying(255)</code>			
<code>full_name</code>	<code>character varying(255)</code>			
<code>given_name</code>	<code>character varying(255)</code>			
<code>google_sub</code>	<code>character varying(255)</code>			
<code>hosted_domain</code>	<code>character varying(255)</code>			
<code>locale</code>	<code>character varying(255)</code>			
<code>picture_url</code>	<code>character varying(255)</code>			
<code>alias</code>	<code>character varying(255)</code>			
<code>status</code>	<code>character varying(255)</code>			
<code>alias_pending</code>	<code>character varying(255)</code>			
<code>date_approved</code>	<code>timestamp without time zone</code>			

Indexes:

"constraint\_4" PRIMARY KEY, btree (id)

### 2. Added “Reviews” database table + backend api endpoints

Fields used to describe a single user review:

Column	Type	Table "public.reviews"	Collation	Nullable	Default
id	bigint			not null	generated by default as identity
reviewer_id	bigint				
item_id	bigint				
date_served	timestamp without time zone				
stars	bigint				
review_text	character varying(255)				
status	character varying(255)				
mod_id	bigint				
mod_comments	character varying(255)				
created_date	timestamp without time zone				
last_edited_date	timestamp without time zone				

Indexes:  
 "reviews\_pk" PRIMARY KEY, btree (id)

Controller endpoints to insert/edit/approve/get reviews:

Reviews			^
PUT	/api/reviews/reviewer	Update a single review	▼
PUT	/api/reviews/moderator	Moderate a single review	▼
POST	/api/reviews/post	Create a new review	▼
GET	/api/reviews	List all reviews created by a specific user	▼
GET	/api/reviews/needsmoderation	List all reviews needing moderation	▼
GET	/api/reviews/all	List all reviews	▼

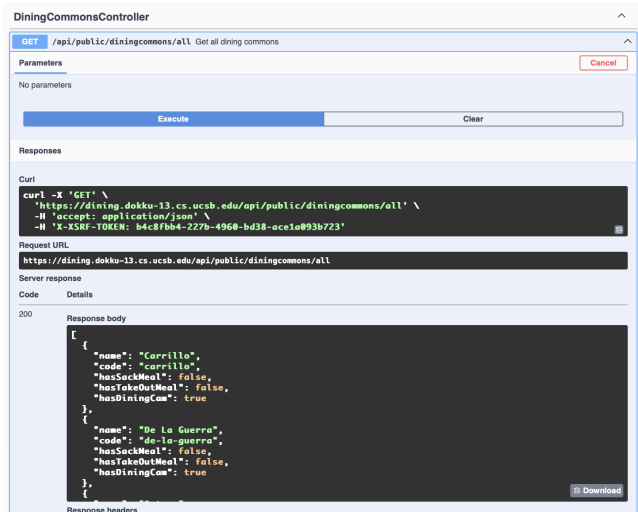
- Regular users can create reviews, update and view their own reviews.
- Admins can additionally list all reviews and reviews needing moderation, and they can approve/reject reviews.

3. Create a service file for get all dining commons:

- Created a model/entity for attributes related to get all API from <https://developer.ucsb.edu/apis/dining/dining-commons>. This is then used for the service file.
- Created a service file to use the API key to access the APIs found in the <https://developer.ucsb.edu/apis/dining/dining-commons>. The service file allows other files to access the APIs found in the link. It uses the given API key to unlock the access to the APIs given to us.

4. Create a controller file for get all dining commons end point:

- Created a endpoint which outputs all current dining commons and its related attributes



## 5. Create a fixtures file for Dining Commons

- The fixture file provides sample data in JSON format, representing Dining Commons objects and attributes as returned by the GET API endpoint

## 6. Created Table, Story Component for Dining Commons

- Have implemented the table component and tests for the table component for presenting the Dining Commons data which contains all the necessary attributes in the columns.
- Created a storybook component for dining commons to visualize testing and development of the user interface

## 7. Create a service file for DiningMenuAPI

- Accesses the dining menu API using the given API key to retrieve relevant information about meal services

## 8. Create a controller file for DiningMenuAPI

- Create an endpoint which outputs all the days meal services are running
- Create an endpoint which outputs which commons are serving meals on the given day
- Create an endpoint which outputs which meals are being served on the given day at the given commons

## **Future Improvements:**

- Create DELETE endpoint, allowing admins to delete any review.
- Use item\_id field to join menu item information alongside the review.
- Integrate unused endpoints into frontend user interface
- Create a GET endpoint, which outputs the menu items being served on the given day at the given commons at the given meal time