CSC309 Project Phase 2 Documentation

Axel Visan (visanaxe)
Danyal Ilyas (ilyasdan)
Chun-Kai Chen (chenc340)

Table of Contents

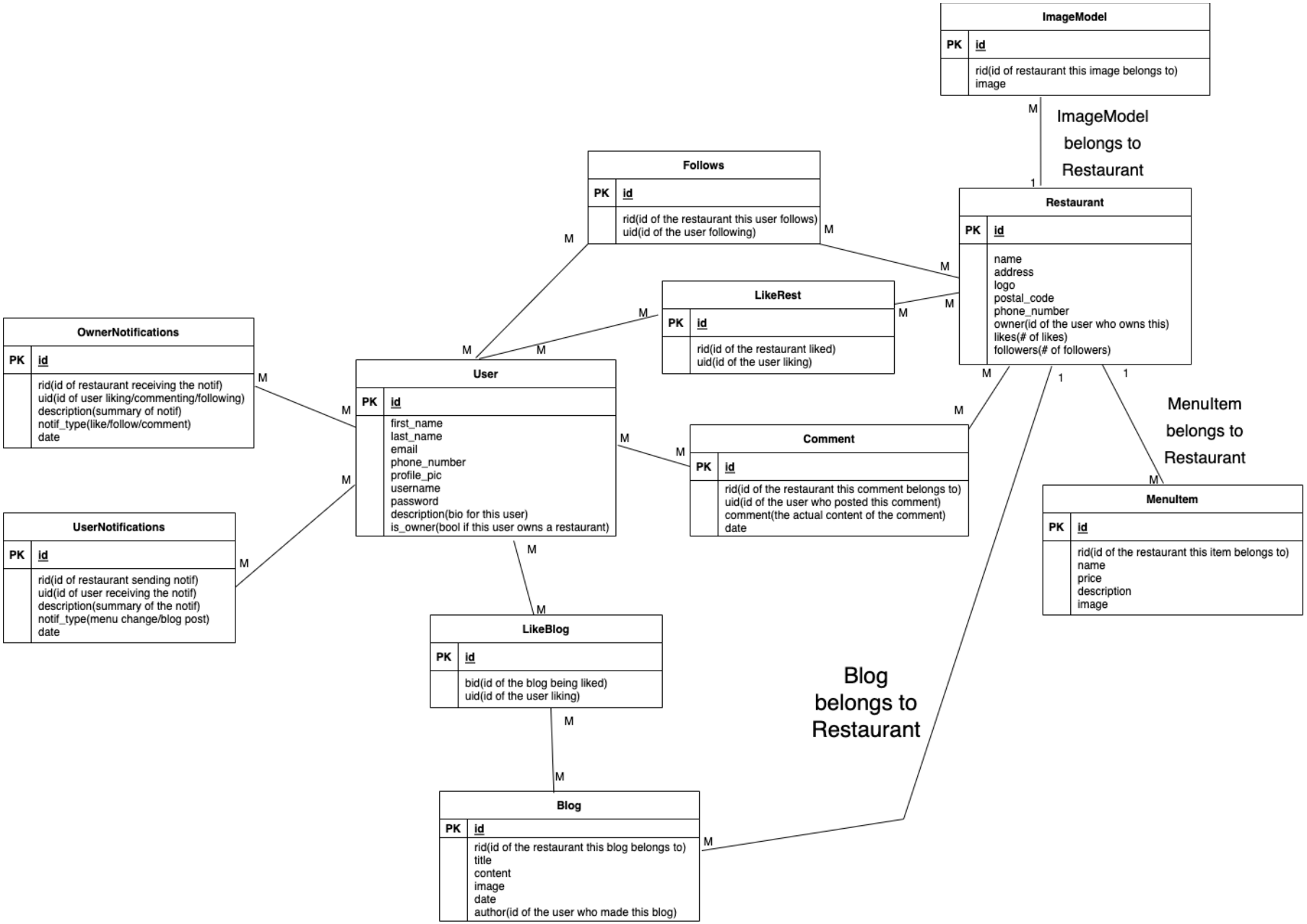
Project Structure

Please take a look at our ERD too. Our project is broken down into several apps listed below. Each app contains several Django python files. Here are the relevant ones:

- views.py: Contains all the views of each app. Our views inherit Django's APIView, DetroyAPIView, CreateAPIView, RetrieveAPIView, ListAPIView... etc Classes define attributes such as the serializer used, the permission classes (ex authenticated), the queryset and so on. These classes also override methods such as GET, POST, PATCH, DELETE and so on. We used various imports, mainly the REST framework.
- urls.py: Each app has its own URL patterns where we define the endpoint URLs within each app (name).
- serializers.py: each app also contains its own serializers for building web APIs. Please see them in the chart below.
- Models.py: Finally, each app also contains its models for our SQL DB.

In our p2_project folder, we have several other files. In settings.py, you will find our access token lifetime to be set to an hour. We decided this is a reasonable which helped us with our testing and which can also be tailored to the best balance between user security and experience. Our urls.py at this level define the path to our apps including the admin.
When users are authenticated, we have quick access to their attributes. We also access specific endpoints with GET requests by specifying PK's / ID's in the URL. POST requests are primarily done through the body section.

Entity Relation Diagram (ERD)



List of Apps and Models

List of Apps	Design of Models (see ERD also)
Blog → Handles the blogs such as creating, viewing, editing and deleting blogs. We use 2 serializers: one for BlogView (latter 3) and another for BlogRegister (creating). The distinction between the 2 is that BlogRegister doesn't contain author, likes and restaurant id since these are either retrieved from the URL, authenticated user profile or default settings.	We have on model 'Blog' with the following fields ● Title of the blog (max 200) ● The blog's text content ● The number of likes, starting at 0 ● An image ● The date and time when the blog was made
Notifications → Handles the delivery of notifications to the users and restaurant owners. We have a serializer for each: ● User notifications contain an id, restaurant id and a description (For example "McDonal's added the Big Mac to their menu") ● Owner notifications contain an id, a user id and a description (for example "John followed your restaurant!")	We made 'Owner Notifications' and 'User notifications', which both have the following fields ● User ID / Owner ID ● Restaurant ID (which they own or follow) ● Notification type ○ User: like / follow / comment ○ Owner: menu change / new item / blog added ● Description of the notification
Restaurant → handles all functionalities of restaurants such as viewing menu, add menu item, editing menu item, viewing restaurant page, creating restaurants, editing restaurants, viewing restaurant comments, searching for restaurants, adding/removing images, viewing images and viewing restaurant blogs. We have several serializers: ● MenuItem ○ Name, price, description, image and restaurant ● RestaurantView ○ Name, address, logo, postal code, phone	We made several models here: Restaurant ● Name ● Address ● Image logo ● Postal code ● Phone number ● The owner of the restaurant (FK to MyUser model) ● Number of likes, starting at 0 ● Number of followers, starting at 0 ImageModel ● Main image field ● Restaurant (FK to the restaurant model)

number and its owner
• AddRestaurant
  ○ Same as RestaurantView, but without the owner field
• Comment
  ○ Used, restaurant commented on, comment content and the date/time posted
• Restaurant
  ○ Name, address, logo, postal code, phone number and its owner
• Image
  ○ Image and its restaurant
	MenuItem
	• Name of the item
	• Price (float)
	• Description
	• Image of the item
	• Restaurant it belongs to (FK to the restaurant model)
	
	Comment → User comments on a restaurant's many-to-many relationship.
	• Comment itself (max length at 400, similar to Twitter)
	• Date and time posted
	• Restaurant it belongs to (FK to the restaurant model)
	• Author (FK to the MyUser model)

Social → Handles all the social interactions between users and/or restaurants. These include comments, follows and the feed. We are only using one serializer for following. All other work is done on the backend.	There are several models here as well Follows (user follows restaurant), LikeBlog (liking a blog) and ListRest (liking a restaurant). These all contain: • Restaurant (FK to the restaurant model) • User (FK to the MyUser model) These models all define relationships between users and restaurants with many-to-many.
Users → Handles everything related to user accounts. We have a serializer for profile (username, profile_pic, first & last name, email and phone #) and another for the signup feature (which has additional password attributes → not visible to the users). • MyUser ○ Username, email, password & password2, description, profile photo, owner status	We defined our MyUser model which extends Django's AbstractUser Model. We added the following fields: • Description (like an about section / bio) • Profile picture • Owner status (Boolean to indicate if User owns a restaurant)

End Points

Create a blog

Description: Create a blog for a restaurant where the restaurant is found by getting the restaurant that belongs to the logged-in user.

Endpoint: http://127.0.0.1:8000/blog/add/

Method: POST

Authenticated: Yes

Status codes: 201 created on success, 403 on user does not own a restaurant, 401 on the user not authenticated, 400 bad request on missing fields

Body:

```
{
        title: <string>
        Content: <string>
        image: <image upload>
}
```

Sample Response:


```
{
  "title": "NEW BLOG! pt67",
  "content": "FIRST!",
  "image": "http://127.0.0.1:8000/Screen_Shot_2022-03-15_at_2.32.18_AM_D9vM6jj.png",
  "rid": 2,
  "likes": 0,
  "author": 2,
  "date": "2022-03-16T02:36:55.984984Z"
}
```

View a Single blog

Description: View a single blog by passing in the blog_id to the url.

Endpoint: http://127.0.0.1:8000/blog/blog_id/view/

Method: GET

Authenticated: No

Status code: 200 on success, 404 on Blog does not exist

Body: N/A

Sample Response:


```
{
  "title": "NEW BLOG!",
  "content": "FIRST!",
  "image": "http://127.0.0.1:8000/Screen_Shot_2022-03-15_at_2.32.18_AM_CVdKMmD.png",
  "rid": 2,
  "likes": 0,
  "author": 2,
  "date": "2022-03-16T02:34:42.712713Z"
}
```


Edit a blog

Description: Edit a blog by passing the blog id into the URL, User must be the owner of the restaurant to which this blog belongs to.

Endpoint: http://127.0.0.1:8000/blog/<blog_id>/edit/

Method: PATCH

Authenticated: YES

Status code: 200 on success, 404 on Blog does not exist, 401 on the user not logged in. 403 user does not own the restaurant to which this blog belongs

Body:

```
{
        title: <string>
        Content: <string>
        image: <image upload>
}
```

Sample response


```
{
  "title": "HELLO",
  "content": "YOU ARE HACKED",
  "image": "http://127.0.0.1:8000/Screen_Shot_2022-03-14_at_4.31.00_AM_85KZqsl.png"
}
```

Delete a blog

Description: Delete a blog by passing the blog id into the URL, User must be the owner of the restaurant to which this blog belongs to.

Endpoint: http://127.0.0.1:8000/blog/<blog_id>/remove/

Method: DELETE

Authenticated: YES

Status code: 204 no content on successful delete, 404 on Blog does not exist, 401 on the user not logged, 403 user does not own the restaurant to which this blog belongs

Body: N/A

Sample Response

View all Blogs that belong to a restaurant

Endpoint: **http://127.0.0.1:8000/restaurant/2/blogs/**

Method: GET

Authorization: No

Status code: HTTP 200 on success, 404 on restaurant does not exist

Body: N/A

Sample response


```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "title": "HELLO",
      "content": "YOU ARE HACKED",
      "image": "http://127.0.0.1:8000/Screen_Shot_2022-03-14_at_4.31.00_AM_85KZqsl.png",
      "rid": 2,
      "likes": 0,
      "author": 2,
      "date": "2022-03-16T02:36:55.984984Z"
    }
  ]
}
```

Follow a restaurant

Description: An authenticated user can follow a restaurant

Endpoint: **http://127.0.0.1:8000/social/follow/**

Method: POST

Authenticated: Yes

Status code: HTTP 201 on success, 400 on bad request, 401 unauthorized, 404 not found

Body:

{

rid: <restaurant id>

}

Sample response

{

"uid": 2,

"rid": 1

}

Unfollow a restaurant

Description: An authenticated user can unfollow a restaurant

Endpoint: http://127.0.0.1:8000/social/unfollow/1/

Method: DELETE

Authenticated: Yes

Status code: HTTP 204 no content on successful delete, 404 if restaurant is not found, 400 bad request (user doesn't follow restaurant)401 unauthorized (if not logged)

Body: N/A

Sample Response: N/A

Like a restaurant

Description: An authenticated user can like a restaurant, which will increment the number of likes on a restaurant

Endpoint: **http://127.0.0.1:8000/social/like/restaurant/**

Method: POST

Authenticated: Yes

Status code: HTTP 200 on success, 404 is restaurant is not found, 400 on bad request, 401 unauthorized

Body:

{

rid: <restaurant id>

}

Sample Response: N/A

Unlike a restaurant

Description: An authenticated user can unlike a restaurant, which will decrement the number of likes on a restaurant

Endpoint: **http://127.0.0.1:8000/social/unlike/restaurant/1/**

Method: DELETE

Authenticated: Yes

Status code: HTTP 200 on success, 404 is restaurant is not found, 400 on user does not like

Body: N/A

Sample Response: N/A

Like a blog

Description: An authenticated user can like a blog, which will increment the number of likes on a blog

Endpoint: **http://127.0.0.1:8000/social/like/blog/**

Method: POST

Authenticated: Yes

Status code: HTTP 200 on success, 404 is restaurant is not found, 400 on bad request, 401 unauthorized

Body:

{

bid: <blog id>

}

Sample response: N/A

Unlike a blog

Description: An authenticated user can unlike a blog, which will decrement the number of likes on a blog

Endpoint: **http://127.0.0.1:8000/social/unlike/blog/1/**

Method: DELETE

Authenticated: Yes

Status code: HTTP 200 on success, 404 is restaurant is not found, 400 on bad request, 401 unauthorized

Body: N/A

Sample response: N/A

Display feed

Description: An authenticated user can display their blog feed based on the restaurants they follow

Endpoint: http://127.0.0.1:8000/social/feed/

Method: GET

Authenticated: Yes

Status code: HTTP 200 on success, 401 unauthorized

Body: N/A

Sample response

{

"count": 1,

"next": **null**,

"previous": **null**,

"results": [

```
    {
      "title": "NEW BLOG!",
      "content": "FIRST!",
      "image": "http://127.0.0.1:8000/Screen_Shot_2022-03-15_at_2.32.18_AM_UuP6umm.png",
      "rid": 1,
      "likes": 1,
      "author": 1,
      "date": "2022-03-16T02:35:16.899572Z"
    }
  ]
}
```


User Login

Description: An unauthenticated user can login to their account

Endpoint: http://127.0.0.1:8000/users/login/

Method: POST

Authenticated: No

Status code: HTTP 200 on success, 401 unauthorized, 400 on bad request

Body:

```
{
        username: <string>
        password: <password>
}
```

Sample Response


```
{
  "refresh":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTY0NzQ4MDUxNiwiaWF
0IjoxNjQ3Mzk0MTE2LCJqdGkiOiI2OWYxZjVhZDNkZWM0YjFmYjkyYTMxNWZmNmZkMGUzMyIsInVzZXJfaWQiOjN
9.EhejOcbszgqZVrHg81ywGnAsJQRQUqpaLbpTrFDDnAQ",
  "access":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjQ3NDgwNTE2LCJpYXQiOj
E2NDczOTQxMTYsImp0aSI6IjA3ZGVlMjc1M2Y4MzRiZWI4ZTUxNWFkYjA2NzZiNDIzIiwidXNlcl9pZCI6M30.LE_V7-
U0Z42BS3EnLhYgTMnbmjLGSKwQohlnM8rp2jc"
}
```

User Register

Description: Any web user can create an account

Endpoint: http://127.0.0.1:8000/users/register/

Method: POST

Authenticated: No

Status code: HTTP 201 created on success, 400 on bad request (i.e. username is not unique, passwords don't match...)

Body:

{

username: <string>

first_name = <string>

last_name = <string>

profile_pic = <image>

email = <email>

password: <password>

password2: <password>

}

Sample Response


```
{
  "username": "danny2",
  "profile_pic": "/Screen_Shot_2022-03-15_at_2.32.18_AM.png",
  "phone_number": "1231231234",
  "first_name": "danny2",
  "last_name": "danny2",
  "email": "kai@gmail.com"
}
```


User View Profile

Description: An authenticated user can view their profile

Endpoint: http://127.0.0.1:8000/users/profile/

Method: GET

Authenticated: Yes

Status code: HTTP 200 on success, 401 unauthorized

Body: N/A

Sample response:


```
{
  "username": "danny2",
  "profile_pic": "http://127.0.0.1:8000/Screen_Shot_2022-03-15_at_2.32.18_AM.png",
  "first_name": "danny2",
  "last_name": "danny2",
```

```
  "email": "kai@gmail.com",
  "phone_number": "1231231234"
}
```


User Edit Profile

Description: An authenticated user can edit their profile

Endpoint: http://127.0.0.1:8000/users/edit/

Method: PATCH

Authenticated: Yes

Status code: HTTP 200 on success, 401 unauthorized, 400 bad request (for example username already taken)

Body:

```
{
        first_name = <string>
        last_name = <string>
        profile_pic = <image>
        email = <email>
}
```

Sample Response


```
{
  "username": "danny2",
  "profile_pic": "http://127.0.0.1:8000/Screen_Shot_2022-03-15_at_2.32.18_AM.png",
  "first_name": "Danyal",
  "last_name": "Ilyas",
  "email": "kai@gmail.com",
  "phone_number": "1231231234"
}
```


User Notifications

Description: An authenticated user can get their notifications

Endpoint: http://127.0.0.1:8000/notifications/user/

Method: GET

Authenticated: Yes

Status code: HTTP 200 on success, 401 unauthorized

Body: N/A

Sample Response:


```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
```

```
    {
        "uid": 2,
        "rid": 1,
        "notif_type": "n",
        "description": "we are gentlemen: Menu item #3 was added!"
    },
    {
        "uid": 2,
        "rid": 1,
        "notif_type": "b",
        "description": "New blog! blog was posted for we are gentlemen!"
    }
  ]
}
```


Owner Notifications

Description: An authenticated user can get their notifications of the restaurants they own

Endpoint: http://127.0.0.1:8000/notifications/owner/

Method: GET

Authenticated: Yes

Status code: HTTP 200 on success, 401 unauthorized

Body: N/A

Sample response


```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
        "id": 10,
        "uid": 4,
        "description": "danny3 liked your restaurant: not gentlemen we r roughmen"
    },
    {
        "id": 8,
        "uid": 4,
        "description": "danny3 liked your blog: HELLO"
```

```
    }
  ]
}
```


Restaurant Menu

Description: Anyone can view a restaurant's menu page

Endpoint: http://127.0.0.1:8000/restaurant/<restaurant_id>/menu/

Method: GET

Authenticated: No

Status code: HTTP 200 on success, 404 on restaurant not found

Body: N/A

Sample response


```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "name": "Root Beer",
      "price": 9.0,
      "description": "get lit",
      "image": "http://127.0.0.1:8000/Screen_Shot_2022-03-14_at_4.31.00_AM.png",
      "rid": 1
    }
  ]
}
```


Add item to restaurant menu

Description: An ower can add to a restaurant's menu page

Endpoint: http://127.0.0.1:8000/restaurant/<restaurant_id>/menu/add/

Method: POST

Authenticated: Yes

Status code: HTTP 201 created on success, 400 bad request, 404 on restaurant not found, 403 forbidden, 401 unauthorized

Body:

{

Name = <string>

Price = <float>

Description = <string>
Image = <image>
}
Sample response: N/A

Edit restaurant item on menu

Description: An owner can edit a restaurant's menu page
Endpoint: http://127.0.0.1:8000/restaurant/menu/1/edit/
Method: PATCH
Authenticated: Yes
Status code: HTTP 200 on success, 400 bad request, 404 on restaurant/item not found, 403 forbidden, 401 unauthorized
Body:
{
Name = <string>
Price = <float>
Description = <string>
Image = <image>
}
Sample response


```
{
  "name": "Root Beer",
  "price": 9.0,
  "description": "get lit",
  "image": "http://127.0.0.1:8000/Screen_Shot_2022-03-14_at_4.31.00_AM.png",
  "rid": 1
}
```


View a restaurant

Description: Anyone can view a restaurant's page
Endpoint: http://127.0.0.1:8000/restaurant/view/<restaurant_id>/
Method: GET
Authenticated: No
Status code: HTTP 200 on success,, 404 on restaurant not found
Body: N/A
Sample response


```
{
  "name": "temp",
  "address": "diner road",
  "logo": "http://127.0.0.1:8000/Screen_Shot_2022-03-15_at_2.32.18_AM_23pkd9L.png",
```

```
  "likes": 0,
  "followers": 0,
  "postal_code": "123 123",
  "phone_number": "1231231234",
  "owner": 1
}
```


Add a restaurant

Description: An authenticated user can create a restaurant

Endpoint: http://127.0.0.1:8000/restaurant/add/

Method: POST

Authenticated: Yes

Status code: HTTP 201 created on success, 400 bad request (for example user already owns restaurant), 401 unauthorized (if not logged in)

Body:

```
{
        name = <string>
        address = <string>
        logo = <image>
        postal_code = <string>
        phone_number = <int>

}
```

Sample Response:


```
{
  "name": "temp",
  "address": "diner road",
  "logo": "http://127.0.0.1:8000/Screen_Shot_2022-03-15_at_2.32.18_AM_23pkd9L.png",
  "likes": 0,
  "followers": 0,
  "postal_code": "123 123",
  "phone_number": "9057780101",
  "owner": 1
}
```


Edit a restaurant

Description: A restaurant owner can edit their profile

Endpoint: http://127.0.0.1:8000/restaurant/edit/<restaurant_id>/

Method: PATCH
Authenticated: Yes
Status code: HTTP 200 on success, 401 unauthorized (if not logged / wrong JWT token), 403 forbidden (example user is not the owner)
Body:
{
name = <string>
address = <string>
logo = <image>
postal_code = <string>
phone_number = <int>

}
Sample response

```
{
  "name": "We are gentlemen",
  "address": "Diner Road",
  "logo": "http://127.0.0.1:8000/daback_Dje6qdA.jpg",
  "likes": 0,
  "followers": 0,
  "postal_code": "123 456",
  "phone_number": "1231231234",
  "owner": 1
}
```


Comment on a restaurant

Description: An authenticated user can comment on a restaurant page
Endpoint: http://127.0.0.1:8000/restaurant/<restaurant_id>/comment/
Method: POST
Authenticated: Yes
Status code: HTTP 201 on success, 400 bad request (for example missing parameters), 404 not found, 401 unauthorized (if not logged / invalid JWT token)
Body:
{
comment = <string>
}
Sample response:

```
{
  "uid": 1,
```

```
  "rid": 1,
  "comment": "I dunno what is going on",
  "date": "2022-03-16T01:53:01.855288Z"
}
```


View restaurant comments

Description: Anyone can view restaurant comments

Endpoint: http://127.0.0.1:8000/restaurant/<restaurant_id>/view/comments

Method: GET

Authenticated: No

Status code: HTTP 200 on success, 404 not found

Body: N/A

Sample response


```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "username": "danny1",
      "restaurant": "we are gentlemen",
      "comment": "I dunno what is going on3"
    },
    {
      "username": "admin",
      "restaurant": "we are gentlemen",
      "comment": "I dunno what is going on2"
    }
  ]
}
```

Search restaurants

Description: Anyone can search a restaurant by address, name or menu item

Endpoint: **http://127.0.0.1:8000/restaurant/search/?query=<value>**

Method: POST

Authenticated: Yes

Status code: HTTP 200 on success, 400 bad request

Parameters:

{

query = <string>

}

Body: N/A

Sample response


```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "name": "we are gentlemen",
      "address": "diner road",
      "logo": "http://127.0.0.1:8000/daback_i9PQ1EQ.jpg",
      "likes": 500,
      "followers": 0,
      "postal_code": "123 456",
      "phone_number": "1231231234",
      "owner": 1
    },
    {
      "name": "Not gentlemen we are roughmen",
      "address": "diner road",
      "logo": "http://127.0.0.1:8000/Screen_Shot_2022-03-15_at_2.32.18_AM_nMVocVd.png",
      "likes": 0,
      "followers": 0,
      "postal_code": "123 123",
      "phone_number": "1231231234",
      "owner": 2
    }
```

```
  ]
}
```


Add image to restaurant

Description: An owner can add an image to their restaurant

Endpoint: http://127.0.0.1:8000/restaurant/<restaurant_id>/add/image/

Method: POST

Authenticated: Yes

Status code: HTTP 201 created on success, 400 bad request (for example missing file), 404 not found, 403 forbidden, 401 unauthorized

Body:


```
{
        image = <image>
}
```


Sample response


```
{
  "image": "http://127.0.0.1:8000/Screen_Shot_2022-03-15_at_2.32.18_AM_2sOnt7D.png",
  "rid": 1
}
```


Remove image from restaurant

Description: An owner can remove an image from their restaurant

Endpoint: http://127.0.0.1:8000/restaurant/remove/<image_id>/image/

Method: DELETE

Authenticated: Yes

Status code: HTTP 204 no content on successfully delete, 404 not found, 403 forbidden, 401 unauthorized

Body: N/A

Sample response: N/A

View restaurant images

Description: Anyone can view a restaurant image

Endpoint: http://127.0.0.1:8000/restaurant/<restaurant_id>/view/images/

Method: GET

Authenticated: No

Status code: HTTP 200 on success, 404 not found

Body: N/A

Sample Response:


```
{
  "count": 2,
```

```
  "next": "http://127.0.0.1:8000/restaurant/1/view/images/?limit=5&offset=5",
  "previous": null,
  "results": [
    {
      "image": "http://127.0.0.1:8000/Screen_Shot_2022-03-15_at_2.32.18_AM_0U6UcnP.png",
      "rid": 1
    },
    {
      "image": "http://127.0.0.1:8000/Screen_Shot_2022-03-15_at_2.32.18_AM_bKdWg1P.png",
      "rid": 1
    }
  ]
}
```


View restaurant blogs

Description: Anyone can view restaurant blogs

Endpoint: http://127.0.0.1:8000/restaurant/<restaurant_id>/blogs/

Method: GET

Authenticated: No

Status code: HTTP 200 on success, 400 bad request, 404 not found

Body: N/A