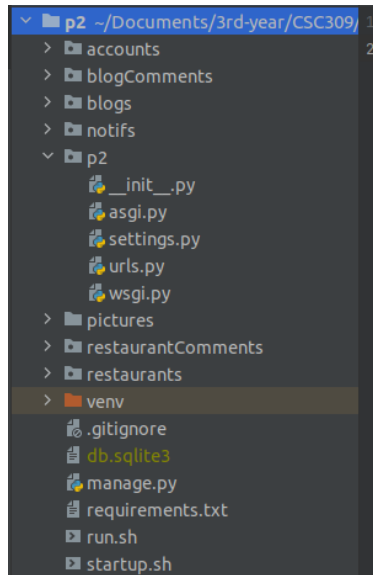


Project Structure

All our app endpoints are found in the folder p2 in urls.py. The virtual env is also contained within the p2 folder(not “P2”). Creating the project is all handled by the startup and run scripts. We have folder for each specific app(eg. blogs) and the urls.py within each app folder contains the specific endpoints for that app. In the following picture we can see the different apps and the overall file structure of our project/



To create and run the project please use the command “source ./startup.sh” and then “./run.sh”. The reason why we have to use the source keyword is to activate the virtual env. If these commands do not work on your system for some reason, it may be a permission issue, so use the keyword “sudo” before the command.

Model Designs

Profile

This is the model we use to “extend” the default User model in django. The fields include:

- `user = models.OneToOneField(User, on_delete=models.CASCADE)`
- `avatar = models.ImageField(null=True, blank=True, upload_to='pictures')`
- `phone_number = models.CharField(null=True, blank=True, max_length=10)`

Here the user field is a onetoone field to a default User object. The reason for this is that it allows us leverage the User object django provided while also adding some extra fields relevant for our project

Blog

This is the model for the blog features. It contains the following fields:

- `restaurant = models.ForeignKey(to=Restaurant, on_delete=models.CASCADE, null=False)`
- `title = models.CharField(max_length=200)`
- `main_image = models.ImageField(upload_to='pictures')`
- `intro = models.TextField()`
- `heading1 = models.CharField(max_length=200)`
- `para1 = models.TextField()`
- `section_image1 = models.ImageField(null=True, blank=True, upload_to='pictures')`
- `heading2 = models.CharField(max_length=200, null=True)`
- `para2 = models.TextField(null=True)`
- `section_image2 = models.ImageField(null=True, blank=True, upload_to='pictures')`
- `conclusion = models.TextField()`
- `created_at = models.DateTimeField(auto_now_add=True)`
- `updated_at = models.DateTimeField(auto_now=True)`

There are some options given however the overall format is fixed(can't have more than 2 sections, heading can be over 200 characters, etc). We also have a field for the creation and update time for the blog object. This is leveraged by the feed feature where we want to sort by recency

BlogLike

This model is used to represent a like to a blog. It includes the following fields:

- `user = models.ForeignKey(to=User, on_delete=models.CASCADE, null=False)`
- `blog_liked = models.ForeignKey(to=Blog, on_delete=models.CASCADE, null=False)`
- `created_at = models.DateTimeField(auto_now_add=True)`

Notice here the `on_delete` params for the fields. If the blog or user who liked the blog is deleted, we delete the blog post like as well

BlogComment

This model is used to represent comments under a blog post. It includes the fields:

- `blog = models.ForeignKey(to=Blog, on_delete=models.CASCADE, null=False)`
- `user = models.ForeignKey(to=User, on_delete=models.CASCADE, null=False)`
- `comment = models.CharField(max_length=600)`

Notice here the `on_delete` params for the fields. If the blog or user who commented is deleted, we delete the blog post comment as well

BlogCommentLike

This model is used to represent like for a blog comment: It includes the following fields:

- `user = models.ForeignKey(to=User, on_delete=models.CASCADE, null=False)`
- `commentLiked = models.ForeignKey(to=BlogComment, on_delete=models.CASCADE, null=False)`

Notice the `on_delete` param for the fields. If the comment or the user is deleted, we also delete the blog comment like

Restaurant

The Restaurant model is used to represent a user's restaurant.

Model Fields:

```
user = models.ForeignKey(to=User, on_delete=models.CASCADE, null=False)
name = models.CharField(max_length=200)
address = models.CharField(max_length=200)
postal_code = models.CharField(max_length=6)
website = models.URLField()
phone_number = PhoneNumberField()
description = models.TextField()
logo = models.ImageField()
```

Notes:

- All fields are required and the proper format for the phone number, website and postal code are defined and enforced in the Create/Edit restaurant endpoints
- Although the user field is a foreign key a user is still only able to create at most one restaurant. This rule is enforced in the restaurant creation endpoint.

RestaurantImage

The RestaurantImage model is used to represent an image for a restaurant.

Model Fields:

```
restaurant = models.ForeignKey(Restaurant, related_name='images',
on_delete=models.CASCADE, null=False)
image = models.ImageField()
```

RestaurantMenuItem

The RestaurantMenuItem model is used to represent a menu item for a restaurant.

Notes:

- All fields are required.
- The `on_delete` parameter is set to `CASCADE` which deletes any restaurant images if the associated restaurant is deleted

Model Fields:

```
restaurant = models.ForeignKey(Restaurant, related_name='menu_items',
on_delete=models.CASCADE, null=False)
name = models.CharField(max_length=200)
price = models.FloatField()
description = models.TextField()
```

Notes:

- The on_delete parameter is set to CASCADE which deletes any restaurant menu items if the associated restaurant is deleted
- The full restaurant menu is obtained through querying the table for all menu items belonging to a restaurant. (See the restaurants/<restaurant_id>/menu/view endpoint in the restaurant endpoints below)

RestaurantComment

This is the model used to represent a comment under the restaurant page. It has the following fields:

- `restaurant = models.ForeignKey(to=Restaurant, on_delete=models.CASCADE, null=False)`
- `user = models.ForeignKey(to=User, on_delete=models.CASCADE, null=False)`
- `comment = models.CharField(max_length=600)`

RestaurantCommentLike

This is the model used to represent a like under restaurant comment . It has the following fields:

- `user = models.ForeignKey(to=User, on_delete=models.CASCADE, null=False)`
- `comment_liked = models.ForeignKey(to=RestaurantComment, on_delete=models.CASCADE, null=False)`

Notifications

Notifications is quite a complicated feature, so we have decided to utilize a 3rd party package called django-notifications-hq==1.7.0. It has many fields, and methods that you can read about [here](#).

Accounts Endpoints

General Note: for endpoints where a list is returned, you can use pagination by adding the following at the end of the url: ?limit=x&offset=y, where x and y are some integers

POST /accounts/register/

What it does: This endpoint is to register a user to the restify app

Request

- Url params: None
- Permission: any
- Request body
 - username
 - email
 - phone_number(optional)
 - first_name
 - last_name
 - avatar(optional)
 - password
 - password2

Response:

- 400 bad request: The fields were not entered properly(detailed message returned), username or email is not unique, passwords did not match, email format not correct, number format not correct, etc.
- 201 created: User is created in database

Example Response Fail:

POST

http://localhost:8000/accounts/register/

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> username	dfsd			
<input checked="" type="checkbox"/> password	12345678i12			
<input checked="" type="checkbox"/> password2	12345678i			
<input checked="" type="checkbox"/> email	tester@gmail.comdsdf			
<input checked="" type="checkbox"/> first_name	test			
<input checked="" type="checkbox"/> last_name	er			
<input type="checkbox"/> phone_number	sdfsdf			
Key	Value	Description		

Body

Cookies (1)

Headers (10)

Test Results

Status: 400 Bad Request

Time: 14 ms

Size: 366 B

Save Response

Pretty

Raw

Preview

Visualize

```
{
  "password": ["passwords do not match"]
}
```

Example Response Success:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> username	tester123			
<input checked="" type="checkbox"/> password	12345678!			
<input checked="" type="checkbox"/> password2	12345678!			
<input checked="" type="checkbox"/> email	tester@gmail.com			
<input checked="" type="checkbox"/> first_name	test			
<input checked="" type="checkbox"/> last_name	er			
Key	Value	Description		

Body Cookies (1) Headers (10) Test Results
Status: 201 Created Time: 128 ms Size: 411 B Save Response

Pretty Raw Preview Visualize

```

{
  "username": "tester123",
  "email": "tester@gmail.com",
  "first_name": "test",
  "last_name": "er"
}

```

POST /accounts/login/

What it does: This endpoint is used to login to the restify app by generating a token

Request

- Url params: None
- Permission: any
- Request body
 - username
 - password

Response:

- 401 unauthorized: No account with the given credentials was found
- 400 bad request: a field was not provided in the request body
- 200 ok: User token is generated

Example Response Fail:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> username	tester123			
<input type="checkbox"/> password	12345678!			
Key	Value	Description		

Body Cookies (1) Headers (10) Test Results
Status: 400 Bad Request Time: 14 ms Size: 359 B Save Response

Pretty Raw Preview Visualize

```

{
  "password": ["This field is required."]
}

```

Example Response Success:

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> username	tester123			
<input checked="" type="checkbox"/> password	12345678!			
Key	Value	Description		

Body

Cookies (1)

Headers (10)

Test Results

Status: 200 OK Time: 93 ms Size: 794 B Save Response

Pretty

Raw

Preview

Visualize

```
{  "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2t1b190eXB1IjoicmVmcWZaCiIsImV4cCI6MTY0NzEzNTY2M5wianF0IjoxNjQ3MDQ5MjYxLCJqdGkiOiIzOTk2NDVhOQ3N2Q0NWl0OGI5OWF1NWY0NGZjMnZhNiIsInVzZXJfaWQ1OjJ9.-k6Qy6MhuItpaTflbFU5MQzurs_UBRgZNUU6SG6ghuPw",  "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2t1b190eXB1IjoieWw1ZXNzIiwiaXNwIjoxNjQ3MzEzNTY2M5wianF0IjoxNjQ3MDQ5MjYxLCJqdGkiOiIzOTk2NDVhOQ3N2Q0NWl0OGI5OWF1NWY0NGZjMnZhNiIsInVzZXJfaWQ1OjJ9.qmmCt-6X-6K801fH7o6nWKA1CDdYgqJ3AKUfb2zm8Tw"}
```

GET /accounts/profile/

What it does: This endpoint is used to retrieve the information of the logged in user

Request

- Url params: None
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

- 401 Unauthorized: could not verify that the sender is a logged in user
- 200 ok: token is valid and account info is retrieved

Example Response Fail:

Body	Cookies (1)	Headers (11)	Test Results	Status: 401 Unauthorized	Time: 8 ms	Size: 547 B	Save Response
Pretty	Raw	Preview	Visualize				
<pre>{ "detail": "Given token not valid for any token type", "code": "token_not_valid", "messages": [{ "token_class": "AccessToken", "token_type": "access", "message": "Token is invalid or expired" }] }</pre>							

Example Response Success:

Type	Bearer Token	Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables
The authorization header will be automatically generated when you send the request. Learn more about authorization		Token
		eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2t1b190eXB1IjoicmVmcWZaCiIsImV4cCI6MTY0NzEzNTY2M5wianF0IjoxNjQ3MDQ5MjYxLCJqdGkiOiIzOTk2NDVhOQ3N2Q0NWl0OGI5OWF1NWY0NGZjMnZhNiIsInVzZXJfaWQ1OjJ9.qmmCt-6X-6K801fH7o6nWKA1CDdYgqJ3AKUfb2zm8Tw

Body

Cookies (1)

Headers (10)

Test Results

Status: 200 OK

Time: 14 ms

Size: 462 B

Save Response

Pretty

Raw

Preview

Visualize

```
{
  "id": 2,
  "username": "tester123",
  "email": "tester@gmail.com",
  "first_name": "test",
  "last_name": "er",
  "user_profile": {
    "avatar": null,
    "phone_number": null
  }
}
```

PUT /accounts/editProfile/

What it does: This endpoint is used to edit an existing user's information

Request

- Url params: None
- Permission: Authenticated(provide the token auth)
- Request body
 - username
 - email
 - phone_number(optional)
 - first_name
 - last_name
 - avatar(optional)
- *** If the optional params are not provided or are provided but are blank, they are set to null in the user profile object

Response:

- 400 Bad Request: The fields are missing, not formatted properly, username or email isn't unique, etc
- 401 Unauthorized: could not verify that the sender is a logged in user
- 200 Ok: User object is updated

Example Response Fail:

The screenshot shows a REST client interface with the 'Body' tab selected. The status bar indicates a '400 Bad Request' with a time of 15 ms and a size of 483 B. The response body is displayed in raw format as a JSON object with error messages for each field.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
{
  "username": ["This field is required."],
  "email": ["This field is required."],
  "first_name": ["This field is required."],
  "last_name": ["This field is required."]
}
```

Example Response Success:

The screenshot shows a REST client interface with the 'Body' tab selected. The status bar indicates a '200 OK' response with a time of 15 ms and a size of 483 B. The response body is displayed in raw format as a JSON object with the updated user profile information.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> username	editProfiler	
<input checked="" type="checkbox"/> email	123@gmail.com	
<input checked="" type="checkbox"/> first_name	edit	
<input checked="" type="checkbox"/> last_name	profile	
Key	Value	Description


```
{ "username": "editProfiler", "email": "123@gmail.com", "first_name": "edit", "last_name": "profile" }
```

Blogs Endpoints

POST /blogs/create/

What it does: This endpoint is used to edit an existing user's information

Request

- Url params: None
- Permission: Authenticated(provide the token auth)
- Request body
 - title
 - main_image
 - intro
 - heading1
 - para1
 - section_image1(optional)
 - heading2(optional)
 - para2(optional)
 - section_image2(optional)
 - Conclusion
- Note: You have to create a restaurant for your user before creating blog

Response:

- 400 Bad Request: The fields are missing, not formatted properly, user does not have a restaurant, etc
- 401 unauthorized: could not verify that the sender is a logged in user
- 201 create: Blog has been created and is associated with the user's restaurant

Example Response Fail:

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	title	testing blog create			
<input checked="" type="checkbox"/>	main_image	Select Files			
<input checked="" type="checkbox"/>	intro	abc			
<input checked="" type="checkbox"/>	heading1	abc			
<input checked="" type="checkbox"/>	para1	abc			
<input checked="" type="checkbox"/>	conclusion	testing blog create complete			
	Key	Value	Description		

Body Cookies (1) Headers (10) Test Results Status: 400 Bad Request Time: 16 ms Size: 409 B Save Response

Pretty Raw Preview Visualize

```
{
  "main_image": ["The submitted data was not a file. Check the encoding type on the form."]
}
```

Example Response Success:

none form-data x-www-form-urlencoded raw binary GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	title	testing blog create			
<input checked="" type="checkbox"/>	main_image	avatar.png ×			
<input checked="" type="checkbox"/>	intro	abc			
<input checked="" type="checkbox"/>	heading1	abc			
<input checked="" type="checkbox"/>	para1	abc			
<input checked="" type="checkbox"/>	conclusion	testing blog create complete			
	Key	Value	Description		

Body Cookies (1) Headers (10) Test Results Status: 201 Created Time: 36 ms Size: 574 B Save Response

Pretty Raw Preview Visualize

```
{
  "title": "testing blog create",
  "main_image": "http://localhost:8000/pictures/avatar_gPdFcE.png",
  "intro": "abc",
  "heading1": "abc",
  "para1": "abc",
  "section_image1": null,
  "heading2": null,
  "para2": null,
  "section_image2": null,
  "conclusion": "testing blog create complete"
}
```

DELETE blogs/<blog_id>/delete/

What it does: This endpoint is to delete an existing blog from the database

Request

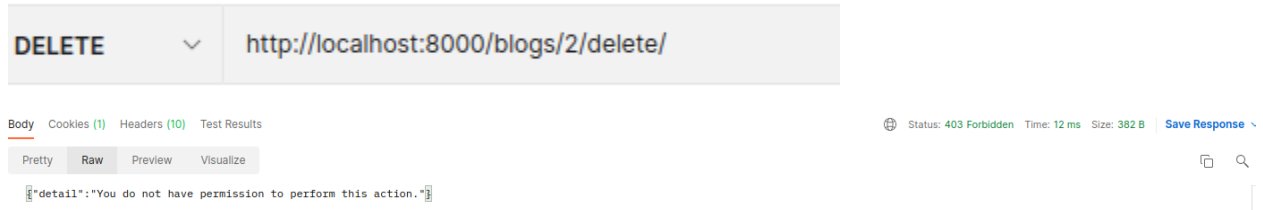
- Url params: blog_id
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

- 404 Not Found: The blog does not exist
- 403 Forbidden: You do not have the permission to delete the blog(you must be the owner of the restaurant for which the blog belongs to)

- 401 Unauthorized: Could not verify that the sender is a logged in user
- 204 no content: Blog has successfully been deleted

Example Response Fail:



DELETE `http://localhost:8000/blogs/2/delete/`


Body Cookies (1) Headers (10) Test Results

Status: 403 Forbidden Time: 12 ms Size: 382 B Save Response

Pretty Raw Preview Visualize

```
{
  "detail": "You do not have permission to perform this action."
}
```

Example Response Success:



Body Cookies (1) Headers (9) Test Results

Status: 204 No Content Time: 18 ms Size: 287 B Save Response

Pretty Raw Preview Visualize

GET blogs/<blog_id>/details/

What it does: This endpoint is to get the details of the specified blog

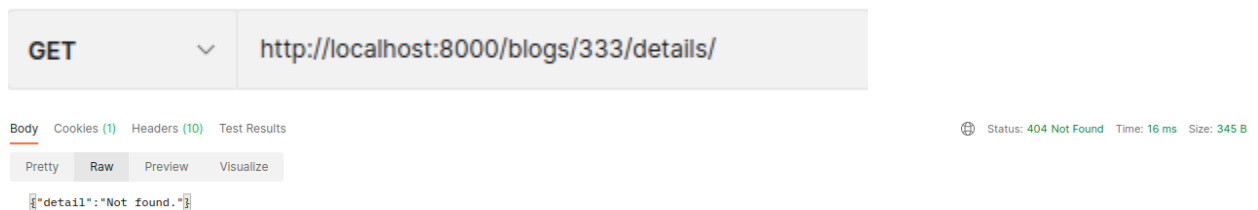
Request

- Url params: blog_id
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

- 404 Not Found: The blog does not exist
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: Blog has successfully been retrieved

Example Response Fail:



GET `http://localhost:8000/blogs/333/details/`

Body Cookies (1) Headers (10) Test Results

Status: 404 Not Found Time: 16 ms Size: 345 B

Pretty Raw Preview Visualize

```
{
  "detail": "Not found."
}
```

Example Response Success:



GET blogs/<restaurant_id>/all/

What it does: This endpoint is to retrieve all the blog posts which are associated with the specified restaurant

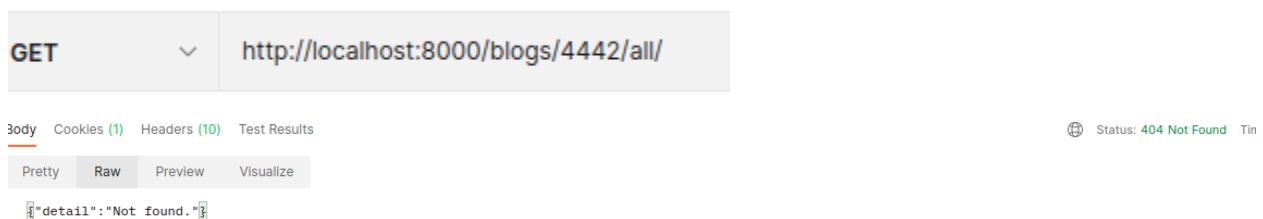
Request

- Url params: restaurant_id
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

- 404 Not Found: The restaurant does not exist
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: list of blogs created by restaurant has successfully been retrieved

Example Response Fail:



Example Response Success:



POST blogs/like/

What it does: This endpoint allows a logged in user to like another blog post

Request

- Permission: Authenticated(provide the token auth)
- Request body
 - blog_liked (this id of the blog which the user want to like)

Response:

- 401 Unauthorized: Could not verify that the sender is a logged in user
- 400 Bad Request: blog_liked not provided, blog_liked not formatted properly, blog_liked is not a valid id for a blog
- 201 Created: blog is liked

Example Response Fail:

POST http://localhost:8000/blogs/like/

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> blog_liked	222	
Key	Text Value	Description

Body Cookies (1) Headers (10) Test Results Status: 400 Bad Request

Pretty Raw Preview Visualize

```
{\"blog_liked\": [\"Invalid pk \\\"222\\\" - object does not exist.\"]}
```

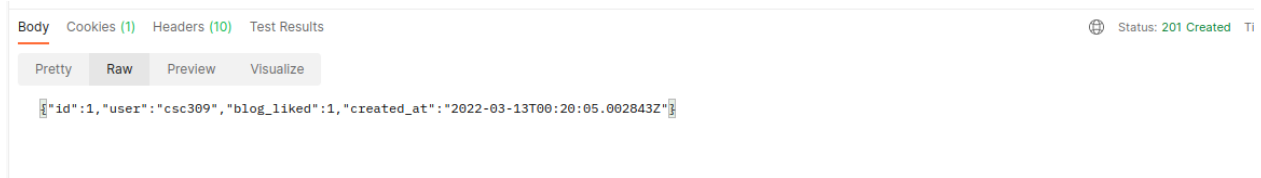
Example Response Success:

POST http://localhost:8000/blogs/like/

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> blog_liked	1
Key	Text Value



DELETE blogs/<blog_id>/unlike/

What it does: This endpoint allows a user to unlike a blog post

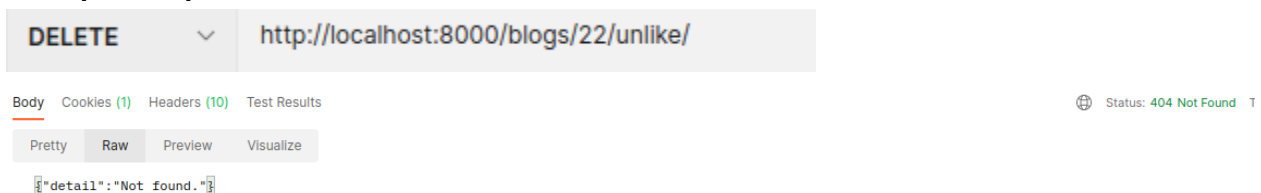
Request

- Url params: blog_id
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

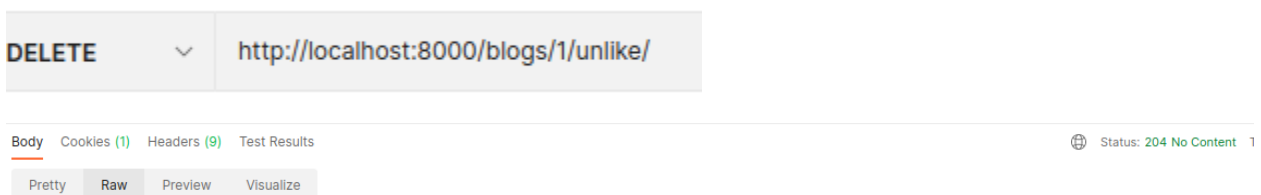
Response:

- 404 Not Found: The blog does not exist
- 403 Forbidden: You do not have permission to perform this action(could be because you never liked the blog so you are not allowed to unlike it)
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 204 No Content: blog has been unliked

Example Response Fail:



Example Response Success:



GET blogs/<blog_id>/likes/amount/

What it does: This endpoint allows a user to get the number of likes that a blog has received

Request

- Url params: blog_id
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

- 404 Not Found: The blog does not exist
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: Number of blog likes has been retrieved

Example Response Fail:

The screenshot shows a REST client interface. The method is GET and the URL is http://localhost:8000/blogs/22/likes/amount/. The status bar indicates a 404 Not Found error. The response body is displayed in raw format as {"detail": "Not found."}.

Example Response Success:

The screenshot shows a REST client interface. The method is GET and the URL is http://localhost:8000/blogs/1/likes/amount/. The status bar indicates a 200 OK status. The response body is displayed in raw format as {"like_count": 1}.

GET blogs/<blog_id>/likes/all/

What it does: This endpoint is to get a list of all the likes to a specific blog post

Request

- Url params: blog_id
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

- 404 Not Found: The blog does not exist
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: All blog likes has been retrieved

Example Response Fail:

GET ⌵ http://localhost:8000/blogs/22/likes/all/

Body Cookies (1) Headers (10) Test Results ⊕ Status: 404 Not Found

Pretty Raw Preview Visualize

```
{\"detail\": \"Not found.\"}
```

Example Response Success:

GET ⌵ http://localhost:8000/blogs/1/likes/all/

Body Cookies (1) Headers (10) Test Results ⊕ Status: 200 OK

Pretty Raw Preview Visualize

```
[{\"id\": 2, \"user\": \"csc309\", \"blog_liked\": 1, \"created_at\": \"2022-03-13T00:31:42.600795Z\"}]
```

GET blogs/feed/

What it does: This endpoint is to get a list of all the likes to a specific blog post, sorted by date created.

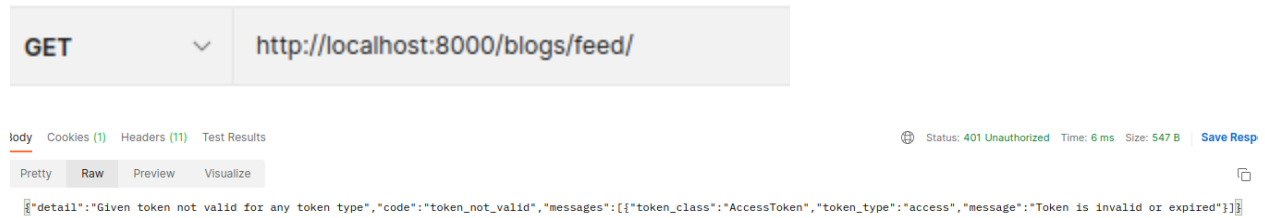
Request

- Url params: N/A
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: All blog posts of restaurants that this user has followed has been retrieved

Example Response Fail:



Example Response Success:



BlogComment Endpoints

POST blogComments/create/

What it does: This endpoint is to create a blog comment for the specified blog

Request

- Url params: N/A
- Permission: Authenticated(provide the token auth)
- Request body
 - blog
 - comment

Response:

- 400 Bad Request: Proper fields were not provided/not formatted correctly,
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 201 Created: Blog comment as been created

Example Response Fail:

POST ▼ <http://localhost:8000/blogComments/create/>

Params Authorization ● Headers (10) **Body** ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL

	KEY	VALUE
<input checked="" type="checkbox"/>	blog	11
<input checked="" type="checkbox"/>	comment	Text ▼
	Key	Value

Body Cookies (1) Headers (10) Test Results 🌐 Status: 400 Bad Request

Pretty **Raw** Preview Visualize

```
{
  "blog": ["Invalid pk \"11\" - object does not exist."],
  "comment": ["This field may not be blank."]
}
```

Example Response Success:

POST ▼ <http://localhost:8000/blogComments/create/>

Params Authorization ● Headers (10) **Body** ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL

	KEY	VALUE
<input checked="" type="checkbox"/>	blog	1
<input checked="" type="checkbox"/>	comment	Text ▼
	Key	Value

Body Cookies (1) Headers (10) Test Results 🌐 Status: 201 Created

Pretty **Raw** Preview Visualize

```
{
  "id": 1, "blog": 1, "user": "csc309", "comment": "wow nice blog bro"
}
```

DELETE blogComments/<blog_comment_id>/delete/

What it does: This endpoint is to delete the specified blog comment


Request

- Url params: blog_comment_id
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

- 404 Not Found: blog comment not found
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 403 Forbidden: You do not have permission to perform this action(could be because the blog comment id is valid but the logged in user is not the one who created it
- 204 No Content: Blog comment is deleted

Example Response Fail:

DELETE  <http://localhost:8000/blogComments/2/delete/>

Body Cookies (1) Headers (10) Test Results

Status: 404 Not Found

Pretty


Raw

Preview

Visualize

```
{
  "detail": "Not found."
}
```

Example Response Success:

DELETE  <http://localhost:8000/blogComments/1/delete/>

Body Cookies (1) Headers (10) Test Results

Status: 403 Forbidden

Pretty

Raw

Preview

Visualize

```
{
  "detail": "You do not have permission to perform this action."
}
```

GET blogComments/<blog_comment_id>/details/

What it does: This endpoint is to get the information about a specific blog comment

Request

- Url params: blog_comment_id
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

- 404 Not Found: blog comment not found
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: Blog comment is retrieved

Example Response Fail:

GET http://localhost:8000/blogComments/22/details/

Body Cookies (1) Headers (10) Test Results

Status: 404 Not Found

Pretty Raw Preview Visualize JSON

```
1  
2 "detail": "Not found."  
3
```

Example Response Success:

GET http://localhost:8000/blogComments/2/details/

Body Cookies (1) Headers (10) Test Results

Status: 200 OK

Pretty Raw Preview Visualize

```
["id":2,"blog":1,"user":"editProfiler","comment":"sdfsdf"]
```

GET blogComments/<blog_id>/all/

What it does: This endpoint gets all the comments to a specified blog

Request

- Url params: blog_id
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

- 404 Not Found: blog not found
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: List of blog comments is retrieved

Example Response Fail:

GET http://localhost:8000/blogComments/asda/all/

Body Cookies (1) Headers (8) Test Results

Status: 404 Not Found

Pretty Raw Preview Visualize

Example Response Success:



POST blogComments/like/

What it does: This endpoint allows a logged in user to like another blog post comment

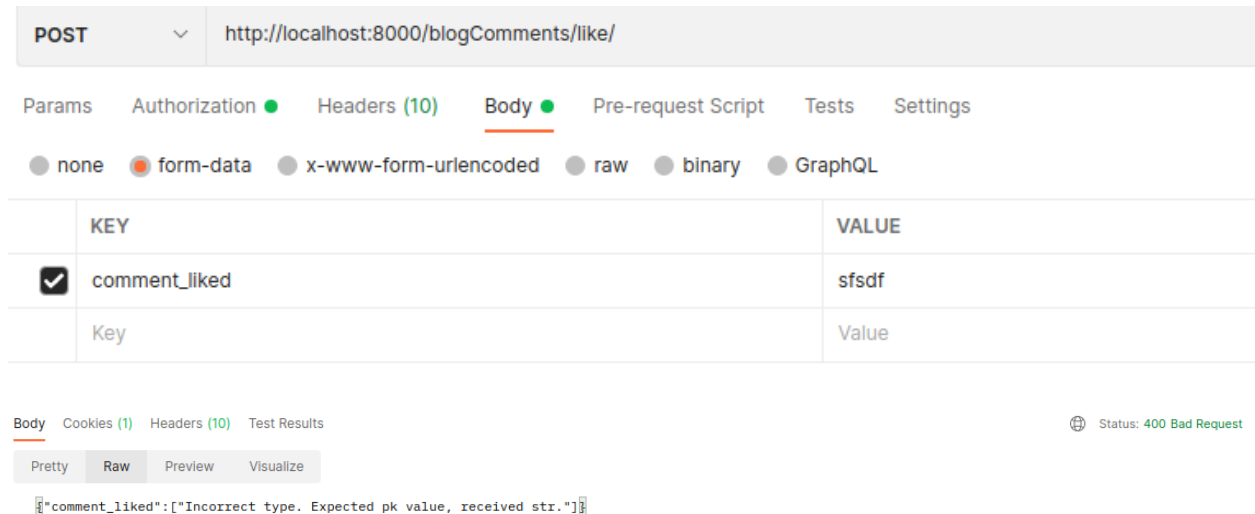
Request

- Permission: Authenticated(provide the token auth)
- Request body
 - comment_liked (this id of the blog comment which the user want to like)

Response:

- 401 Unauthorized: Could not verify that the sender is a logged in user
- 400 Bad Request: comment_liked not provided, comment_liked not formatted properly, is not a valid id for a blog comment, etc
- 201 Created: blog comment is created

Example Response Fail:



Example Response Success:

POST ▼ http://localhost:8000/blogComments/like/

Params Authorization ● Headers (10) Body ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL

	KEY	VALUE
<input checked="" type="checkbox"/>	comment_liked	3
	Key	Text ▼ Value

Body Cookies (1) Headers (10) Test Results 🌐 Status: 201 Created

Pretty Raw Preview Visualize

```
{
  "id": 2,
  "user": "csc309",
  "comment_liked": 3
}
```

DELETE blogs/<blog_comment_id>/unlike/

What it does: This endpoint allows a user to unlike a blog comment

Request

- Url params: blog_comment_id
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

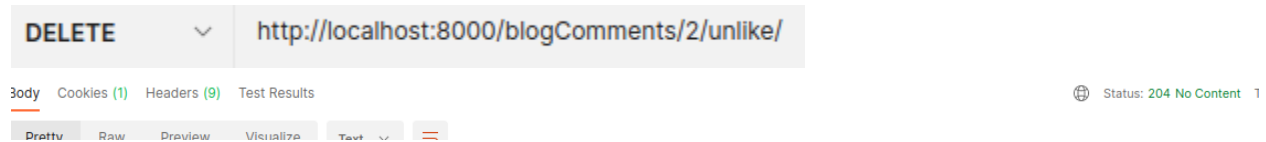
- 404 Not Found: The blog comment does not exist
- 403 Forbidden: You do not have permission to perform this action(could be because you never liked the blog comment so you are not allowed to unlike it)
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 204 No Content: blog comment has been unliked

Example Response Fail:

DELETE ▼ http://localhost:8000/blogComments/sdf/unlike/



Example Response Success:



GET blogs/<blog_comment_id>/likes/all/

What it does: This endpoint is to get a list of all the likes to a specific blog post comment

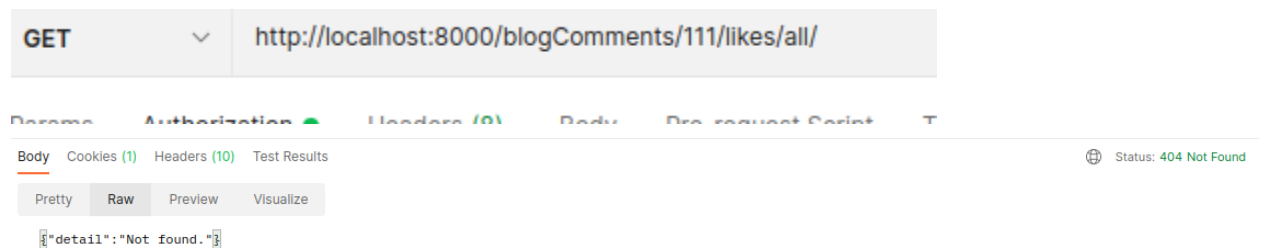
Request

- Url params: blog_comment_id
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

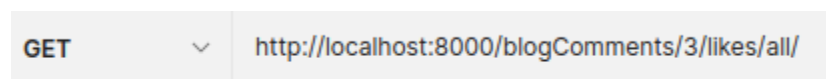
Response:

- 404 Not Found: The blog comment does not exist
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: All blog comment likes has been retrieved

Example Response Fail:



Example Response Success:





GET blogs/<blog_comment_id>/likes/amount/

What it does: This endpoint allows a user to get the number of likes that a blog comment has received

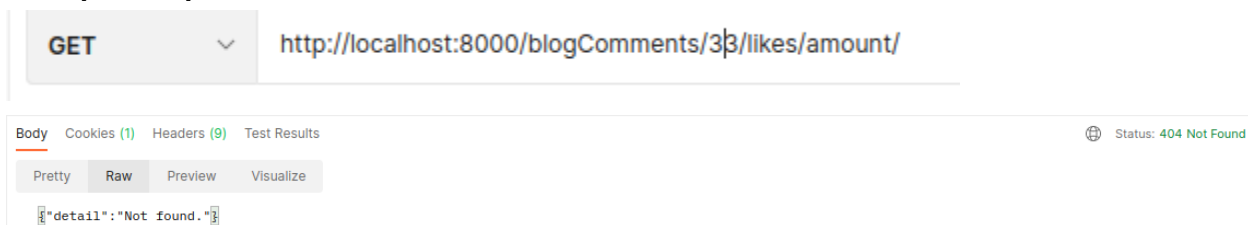
Request

- Url params: blog_comment_id
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

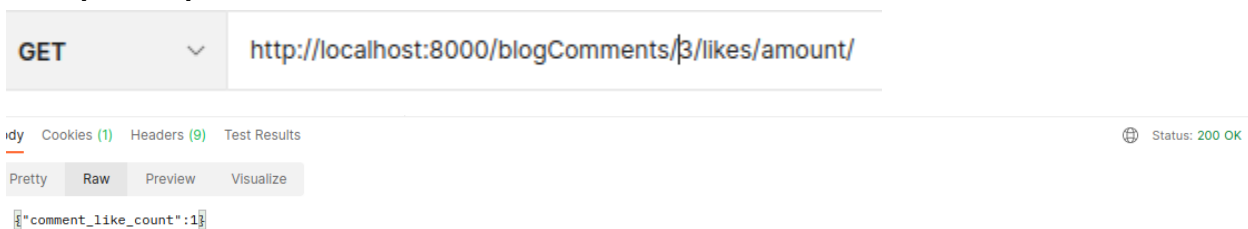
Response:

- 404 Not Found: The blog comment does not exist
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: Number of blog comment likes has been retrieved

Example Response Fail:



Example Response Success:



RestaurantComment Endpoints

***These are the same as BlogComments Endpoints, to use them follow the same instructions as the corresponding BlogComment endpoint but replace the word blog with restaurant wherever you see it

Notifs Endpoints

GET notif/all/

What it does: This endpoint allows user to retrieve all unread notifications

Request

- Url params: N/A
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: all notifications are retrieved

Example Response Fail:

The screenshot shows a REST client interface with a GET request to `http://localhost:8000/notifs/all/`. The response status is 401 Unauthorized. The response body is displayed in raw format, showing a JSON error message: `{ "detail": "Given token not valid for any token type", "code": "token_not_valid", "messages": [{ "token_class": "AccessToken", "token_type": "access", "message": "Token is invalid or expired" }] }`.

Example Response Success:

The screenshot shows a REST client interface with a GET request to `http://localhost:8000/notifs/all/`. The response status is 200 OK. The response body is displayed in raw format, showing a JSON array of notification objects. The first object is: `{ "id": 9, "level": "info", "recipient_id": 4, "unread": true, "actor_content_type_id": 10, "actor_object_id": 4, "verb": "a restaurant you follow has posted a blog", "description": null, "target_content_type_id": null, "target_object_id": null, "action_object_content_type_id": null, "action_object_object_id": null, "timestamp": "2022-03-13T03:07:04.707721Z", "public": true, "deleted": false, "emailed": false, "data": null }`.

POST notif/read_all/

What it does: This endpoint allows user to marks all notifications as read

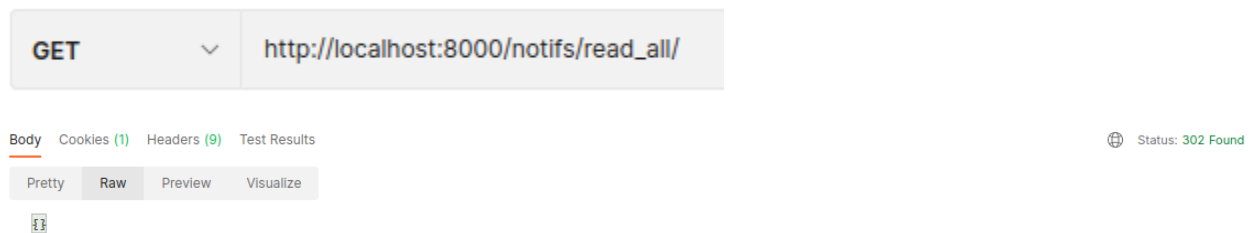
Request

- Url params: N/A
- Permission: Authenticated(provide the token auth)
- Request body
 - N/A

Response:

- 401 Unauthorized: Could not verify that the sender is a logged in user
- 302 Found: all notifications are marked as read

Example Response Success:



Restaurants Endpoints

POST /restaurants/create/

What it does: Creates a restaurant for a user.


Request

- Url params: None
- Permission: Authenticated(provide the token auth)
- Request body
 - name

- address
- post_code
- website
- phone_number
- description
- logo

Response:

- 400 bad request: The fields were not entered properly(detailed message returned), incorrect postal code format, incorrect website format/invalid website, incorrect phone number format, user can only create one restaurant, etc.
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 201 created: Restaurant created in database

POST  http://localhost:8000/restaurants/create/



Example Response Success:

Body   201 Created 160 ms 525 B [Save Response](#)

Pretty Raw Preview Visualize  

```
{
  "name": "Miku",
  "address": "123 street",
  "postal_code": "L5B0C6",
  "website": "https://miku.com",
  "phone_number": "+16045052843",
  "description": "hello",
  "logo": "http://localhost:8000/EoH5XGGWEAUYP RR_yhARWfK.png"
}
```

Example Response Fail:

Body   400 Bad Request 152 ms 404 B

Pretty Raw Preview Visualize

```
{
  "postal_code": ["Please enter a valid 6 character postal code (eg. M5J2R8)"]
}
```

PUT /restaurants/edit/

What it does: Edits a user's restaurant.

Request

- Url params: None
- Permission: Authenticated(provide the token auth)
- Request body
 - name
 - address
 - post_code
 - website
 - phone_number
 - description
 - logo

Response:

- 400 bad request: The fields were not entered properly(detailed message returned), incorrect postal code format, incorrect website format/invalid website, incorrect phone number format, etc.
- 404 Not Found: Restaurant not found. (User has not created a restaurant)
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: Restaurant successfully updated

PUT	▼	http://localhost:8000/restaurants/edit/
-----	---	---

Example Response Success:

Body ▼ 200 OK 159 ms 526 B [Save Response](#)

Pretty Raw Preview Visualize

```
{
  "name": "Miku",
  "address": "123 street",
  "postal_code": "L5B0C6",
  "website": "https://miku.com",
  "phone_number": "+16045052843",
  "description": "hello",
  "logo": "http://localhost:8000/EoH5XGGWEAUYP RR_xGbcGv4.png"
}
```

Example Response Fail:

Body ▼ 404 Not Found

Pretty Raw Preview Visualize

```
{
  "detail": "Not found."
}
```

GET /restaurants/my-restaurant/

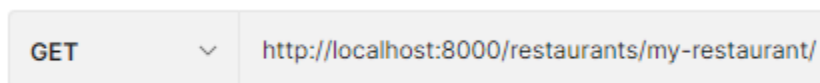
What it does: View a user's restaurant

Request

- Url params: None
- Permission: Authenticated(provide the token auth)
- Request body
 - None

Response:

- 404 Not Found: Restaurant not found. (User has not created a restaurant)
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: Restaurant found and returned



Example Response Success:

Body 200 OK 154 ms 517 B [Save](#)

Pretty Raw Preview Visualize

```
{
  "name": "Miku",
  "address": "123",
  "postal_code": "L5B0C6",
  "website": "https://miku.com",
  "phone_number": "+16045052843",
  "description": "miku",
  "logo": "http://localhost:8000/EoH5XGGWEAUYP RR_VsN0C1q.png"
}
```

Example Response Fail:

Body 404 Not Found

Pretty Raw Preview Visualize

```
{
  "detail": "Not found."
}
```

GET /restaurants/<restaurant_id>/view/

What it does: View info on a restaurant (specified by restaurant_id)

Request

- Url params: Restaurant ID

- Permission: Any
- Request body
 - None

Response:

- 404 Not Found: Restaurant with restaurant_id not found.
- 200 OK: Restaurant found and returned

Example Response Success:



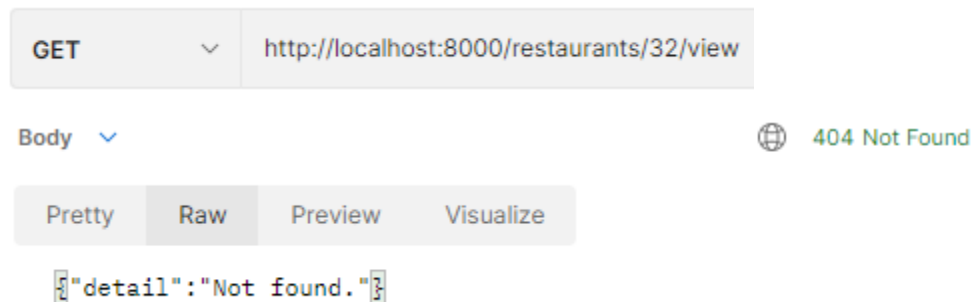
GET ⌵ http://localhost:8000/restaurants/6/view

Body ⌵ 🌐 200 OK 149 ms 517 B [Save](#)

Pretty Raw Preview Visualize

```
{
  "name": "Miku",
  "address": "123",
  "postal_code": "L5B0C6",
  "website": "https://miku.com",
  "phone_number": "+16045052843",
  "description": "miku",
  "logo": "http://localhost:8000/EoH5XGGWEAUYP RR_gj95JWu.png"
}
```

Example Response Fail:



GET ⌵ http://localhost:8000/restaurants/32/view

Body ⌵ 🌐 404 Not Found

Pretty Raw Preview Visualize

```
{
  "detail": "Not found."
}
```

POST /restaurants/follow/


What it does: Follows a restaurant.

Request


- Url params: None
- Permission: Authenticated(provide the token auth)
- Request body
 - Restaurant

Response:

- 400 bad request: Restaurant not provided.
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 201 created: User successfully followed the restaurant

POST  http://localhost:8000/restaurants/follow/

Example Response Success:


Body 

 201 Created 161 ms 362 B

Pretty Raw Preview Visualize

```
{ "id": 1, "restaurant": 7, "user": "yoshio" }
```

Example Response Fail:

Body 

 400 Bad Request 149 ms 369 B

Pretty Raw Preview Visualize

```
{ "restaurant": ["This field is required."] }
```

DELETE /restaurants/<restaurant_id>unfollow/

What it does: Unfollows a restaurant.

Request

- Url params: restaurant_id
- Permission: Authenticated(provide the token auth)
- Request body
 - Restaurant

Response:

- 400 bad request: Restaurant not provided.
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 403 Forbidden: You do not have permission to perform this action. (Cannot unfollow a restaurant that the user is not following)
- 204 No Content: User successfully unfollowed restaurant

Example Response Success:

DELETE

▼

http://localhost:8000/restaurants/7/unfollow/

Body ▼

204 No Content 162 ms 295 B

Pretty

Raw

Preview

Visualize

Example Response Fail:

DELETE

▼

http://localhost:8000/restaurants/7/unfollow/

Body ▼

403 Forbidden 163 ms 390 B

Pretty

Raw

Preview

Visualize

```
{"detail": "You do not have permission to perform this action."}
```

POST /restaurants/like/

What it does: Likes a restaurant.

Request

- Url params: None
- Permission: Authenticated(provide the token auth)
- Request body
 - Restaurant

Response:

- 400 bad request: Restaurant not provided.
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 201 created: User successfully liked the restaurant

POST

▼

http://localhost:8000/restaurants/like/

Example Response Success:

Body ▾

🌐 201 Created 152 ms 362 B

Pretty Raw Preview Visualize

```
{ "id": 1, "user": "yoshio", "restaurant": 7 }
```

Example Response Fail:

Body ▾

🌐 400 Bad Request 150 ms 369 B

Pretty Raw Preview Visualize

```
{ "restaurant": [ "This field is required." ] }
```

DELETE /restaurants/restaurant_id/unlike/

What it does: Unlikes a restaurant.

Request

- Url params: restaurant_id
- Permission: Authenticated(provide the token auth)
- Request body
 - None

Response:

- 401 Unauthorized: Could not verify that the sender is a logged in user
- 403 Forbidden: You do not have permission to perform this action. (Cannot unlike a restaurant that the user has not liked)
- 204 No Content: User successfully unliked restaurant

DELETE ▾

http://localhost:8000/restaurants/7/unlike/

Example Response Success:

Body ▾

🌐 204 No Content 162 ms 295 B

Pretty Raw Preview Visualize

Example Response Fail:

Body 

 403 Forbidden 155 ms 390 B

Pretty Raw Preview Visualize

```
{ "detail": "You do not have permission to perform this action." }
```

POST /restaurants/image/add/

What it does: Add an image to a restaurant

Request

- Url params: None
- Permission: Authenticated(provide the token auth)
- Request body
 - name
 - address
 - post_code
 - website
 - phone_number
 - description
 - logo


Response:

- 400 bad request: Image not provided.
- 404 Not Found: Restaurant not found. (User has not created a restaurant)
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 201 Created: Restaurant Image successfully created.

POST 

http://localhost:8000/restaurants/image/add/

Example Response Success:

Body 

 201 Created

Pretty Raw Preview Visualize

```
{ "image": "http://localhost:8000/EoH5XGGWEAUYP RR_UdjKqgQ.png" }
```

Example Response Fail:

Body ▾

🌐 400 Bad Request

Pretty

Raw

Preview

Visualize

```
{ "image": ["No file was submitted."] }
```

DELETE /restaurants/image/<image_id>/delete/

What it does: Deletes an image for a restaurant

Request

- Url params: image_id
- Permission: Authenticated(provide the token auth)
- Request body
 - None

Response:

- 404 Not Found: Restaurant not found. (User has not created a restaurant), Image with image_id not found.
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 204 No Content: Restaurant Image successfully deleted.

Example Response Success:

DELETE ▾

http://localhost:8000/restaurants/image/3/delete/

Body ▾

🌐 204 No Content 154 ms 295 B

Pretty

Raw

Preview

Visualize

Example Response Fail:

DELETE ▾

http://localhost:8000/restaurants/image/32/delete/

Body ▾

🌐 404 Not Found 149 ms 350 B

Pretty Raw Preview Visualize

```
{ "detail": "Not found." }
```

GET /restaurants/<restaurant_id>/image/all/

What it does: Lists all images for a restaurant

Request

- Url params: restaurant_id
- Permission: None
- Request body
 - None

Response:

- 404 Not Found: Restaurant with restaurant_id not found.
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: Successfully retrieved list of restaurant images

Example Response Success:

GET ▾ http://localhost:8000/restaurants/6/image/all/

Body ▾

🌐 200 OK 150 ms 444 B | [Save](#)

Pretty Raw Preview Visualize

```
[{"image": "http://localhost:8000/EoH5XGGWEAUYP RR_u6QExhU.png"}, {"image": "http://localhost:8000/totororain_ztFLPGb.jpg"}]
```

Example Response Fail:

GET ▾ http://localhost:8000/restaurants/32/image/all/

Body ▾

🌐 404 Not Found 151 ms 353 B

Pretty Raw Preview Visualize

```
{ "detail": "Not found." }
```

POST /restaurants/menu/add/

What it does: Add a menu item to the user's restaurant's menu.

Request

- Url params: None
- Permission: Authenticated(provide the token auth)
- Request body
 - Name
 - Price
 - Description

Response:

- 400 Bad Request: name not provided, price not provided, description not provided.
- 404 Not Found: Restaurant not found. (User has not created a restaurant)
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 201 Created: Successfully created menu item

POST ▾

http://localhost:8000/restaurants/menu/add

Example Response Success:

Body ▾

🌐 201 Created 157 ms 387 B

Pretty Raw Preview Visualize


JSON ▾



```
1 {  
2   "name": "Tuna Roll",  
3   "price": 5.99,  
4   "description": "Roll with Tuna"  
5 }
```

Example Response Fail:

Body ▾ 400 Bad Request 155 ms 364 B

Pretty Raw Preview Visualize JSON ▾ 

```
1 {  
2   "price": [  
3     "This field is required."  
4   ]  
5 }
```

DELETE /restaurants/menu/<menu_item_id>/delete/

What it does: Deletes a menu item from the user's restaurant's menu.

Request

- Url params: menu_item_id
- Permission: Authenticated(provide the token auth)
- Request body
 - None


Response:

- 404 Not Found: Menu item not found, Restaurant not found. (User has not created a restaurant)
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 204 No Content: Successfully deleted menu item

Example Response Success:

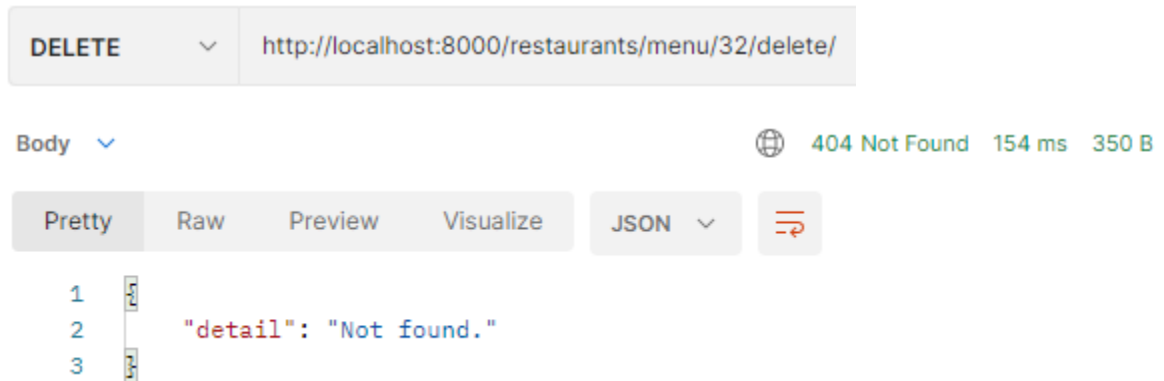
DELETE ▾ <http://localhost:8000/restaurants/menu/11/delete/>

Body ▾ 204 No Content 154 ms 295 B

Pretty Raw Preview Visualize Text ▾ 

```
1
```

Example Response Fail:



PUT /restaurants/menu/<menu_item_id>/edit/

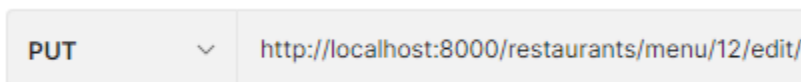
What it does: Edits a menu item from the user's restaurant's menu.

Request

- Url params: menu_item_id
- Permission: Authenticated(provide the token auth)
- Request body
 - Name
 - Price
 - Description

Response:

- 404 Not Found: name not provided, price not provided, description not provided, Menu item not found, Restaurant not found. (User has not created a restaurant)
- 401 Unauthorized: Could not verify that the sender is a logged in user
- 200 OK: Successfully edited menu item



Example Response Success:


Body ▾ 🌐 200 OK 158 ms 388 B

Pretty Raw Preview Visualize JSON ▾ 

```
1 {  
2   "name": "Tuna Roll",  
3   "price": 5.99,  
4   "description": "Roll with Tuna"  
5 }
```

Example Response Fail:

Body ▾ 🌐 400 Bad Request 150 ms 370 B

Pretty Raw Preview Visualize JSON ▾ 

```
1 {  
2   "price": [  
3     "This field is required."  
4   ]  
5 }
```

GET /restaurants/<restaurant_id>/menu/view/

What it does: View a restaurant's menu.

Request

- Url params: restaurant_id
- Permission: None
- Request body
 - None

Response:

- 404 Not Found: Restaurant with restaurant_id not found.
- 200 OK: Successfully retrieved menu

Example Response Success:


GET  http://localhost:8000/restaurants/7/menu/view/



Body   200 OK 151 ms 463 B [Save Response](#)

Pretty Raw Preview Visualize  

```
[{"name": "Tuna Roll", "price": 5.99, "description": "Roll with Tuna"}, {"name": "Dynamite Roll", "price": 9.99, "description": "Roll with dynamite"}]
```

Example Response Fail:

GET  http://localhost:8000/restaurants/32/menu/view/

Body   404 Not Found 150 ms 353 B

Pretty Raw Preview Visualize

```
{"detail": "Not found."}
```

GET /restaurants/search/<str:q>/

What it does: Retrieves restaurants that have <q> as their address, name, or menu item

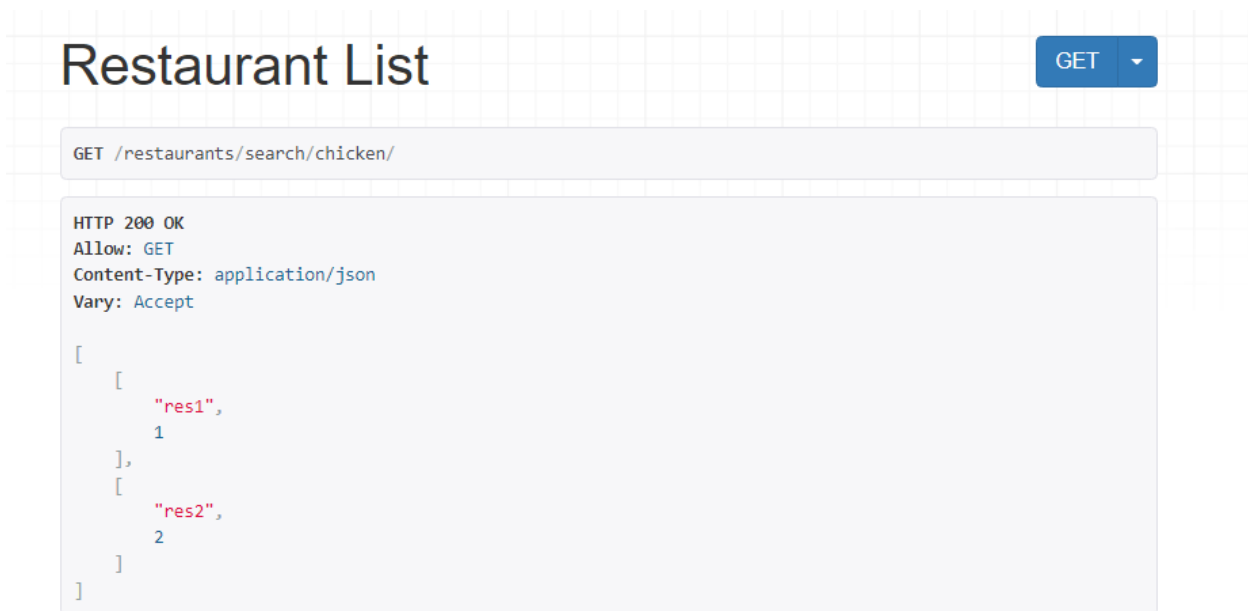
Request

- Url params: <str:q>
- Permission: None
- Request body
 - None

Response:

- 200 OK: Successfully retrieved restaurant
- No fail response, we just get an empty list if no values are retrieved from search

Example Response Success:



GET /restaurants/followed/<int:restaurant_id>/

What it does: Returns a boolean value indicating whether the logged in user has followed the restaurant with the given <restaurant_id>

Request

- Url params: <restaurant_id>
- Permission: Authenticated(provide the token auth)
- Request body
 - None

Response:

- 200 OK: Returns whether the logged in user has followed the restaurant with the given <restaurant_id>
- 404 Not Found: The restaurant does not exist
- 401 Unauthorized: Could not verify that the sender is a logged in user

Example Response Success:

Restaurant Followed

Restaurant Followed

GET

GET /restaurants/followed/1/

HTTP 200 OK
Allow: GET
Content-Type: application/json
Vary: Accept

```
{  
  "Restaurant Followed": "False"  
}
```

Example Response Fail

Restaurant Followed

Restaurant Followed

GET

GET /restaurants/followed/1/

HTTP 401 Unauthorized
Allow: GET
Content-Type: application/json
Vary: Accept
WWW-Authenticate: Bearer realm="api"

```
{  
  "detail": "Authentication credentials were not provided."  
}
```

Restaurant Followed

Restaurant Followed

GET

GET /restaurants/followed/890/

HTTP 404 Not Found
Allow: GET
Content-Type: application/json
Vary: Accept

```
{  
  "detail": "Not found."  
}
```

GET /restaurants/liked/<int:restaurant_id>/

What it does: Returns a boolean value indicating whether the logged in user has liked the restaurant with the given <restaurant_id>

Request

- Url params: <restaurant_id>
- Permission: Authenticated(provide the token auth)
- Request body
 - None

Response:

- 200 OK: Returns whether the logged in user has liked the restaurant with the given <restaurant_id>
- 404 Not Found: The restaurant does not exist
- 401 Unauthorized: Could not verify that the sender is a logged in user

Example Response Success:

The screenshot shows a REST client interface with a grid background. At the top, there's a header 'Restaurant Liked'. Below it, the title 'Restaurant Liked' is displayed in large font, with a 'GET' button to its right. The request bar shows 'GET /restaurants/liked/1/'. The response area displays the following details:

```
HTTP 200 OK
Allow: GET
Content-Type: application/json
Vary: Accept

{
  "Restaurant Liked": "False"
}
```

Example Response Fail:

The screenshot shows a REST client interface with a grid background. The request bar shows 'GET /restaurants/liked/85/'. The response area displays the following details:

```
HTTP 404 Not Found
Allow: GET
Content-Type: application/json
Vary: Accept

{
  "detail": "Not found."
}
```



GET /numberfollowers/<int:restaurant_id>/

What it does: Returns the number of followers the restaurant with the <restaurant_id> has

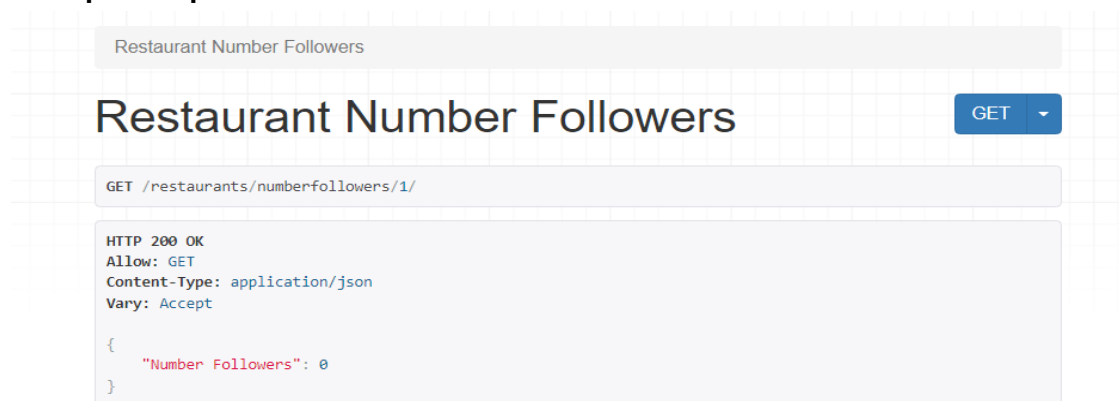
Request

- Url params: <restaurant_id>
- Permission: None
- Request body
 - None

Response:

- 200 OK: Returns the number of followers the restaurant with the <restaurant_id> has
- 404 Not Found: The restaurant does not exist

Example Response Success:



Example Fail Response

Django REST framework

admin

Restaurant Number Followers

Restaurant Number Followers

GET

GET /restaurants/numberfollowers/85/

HTTP 404 Not Found

Allow: GET

Content-Type: application/json

Vary: Accept

```
{
  "detail": "Not found."
}
```

GET /numberlikes/<int:restaurant_id>/

What it does: Returns the number of likes the restaurant with the <restaurant_id> has

Request

- Url params: <restaurant_id>
- Permission: None
- Request body
 - None

Response:

- 200 OK: Returns the number of likes the restaurant with the <restaurant_id>
- 404 Not Found: The restaurant does not exist

Example Response Success:

Restaurant Number Liked

Restaurant Number Liked

GET

GET /restaurants/numberlikes/1/

HTTP 200 OK

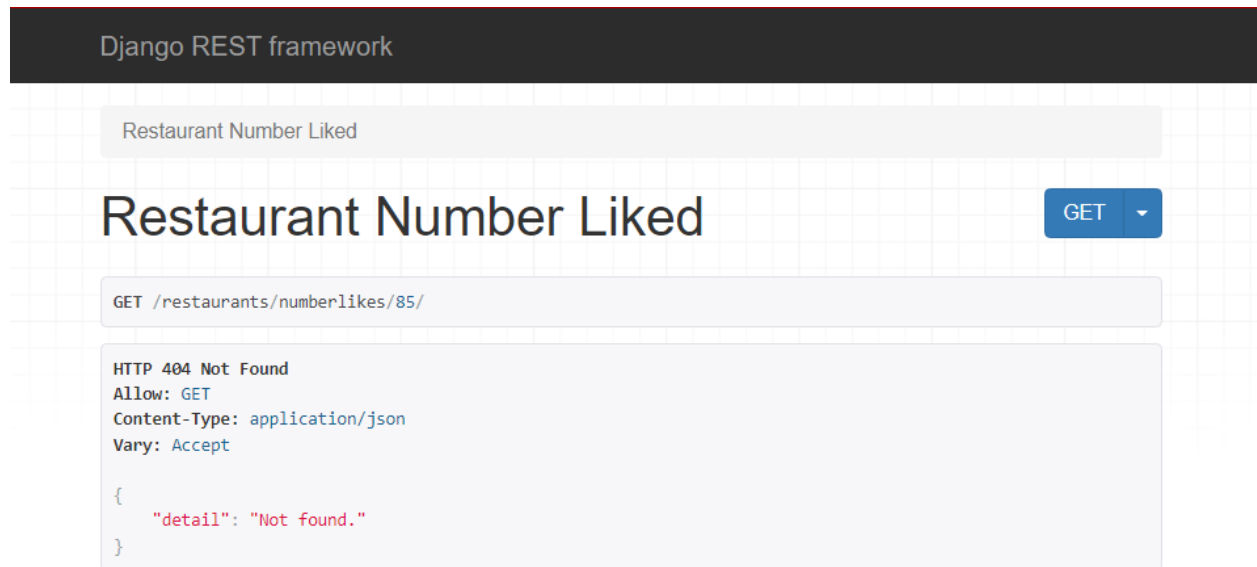
Allow: GET

Content-Type: application/json

Vary: Accept

```
{
  "Number Liked": 1
}
```

Example Response Fail:



GET /images/<int:restaurant_id>/

What it does: Return the images of the restaurant that corresponds to this <restaurant_id>

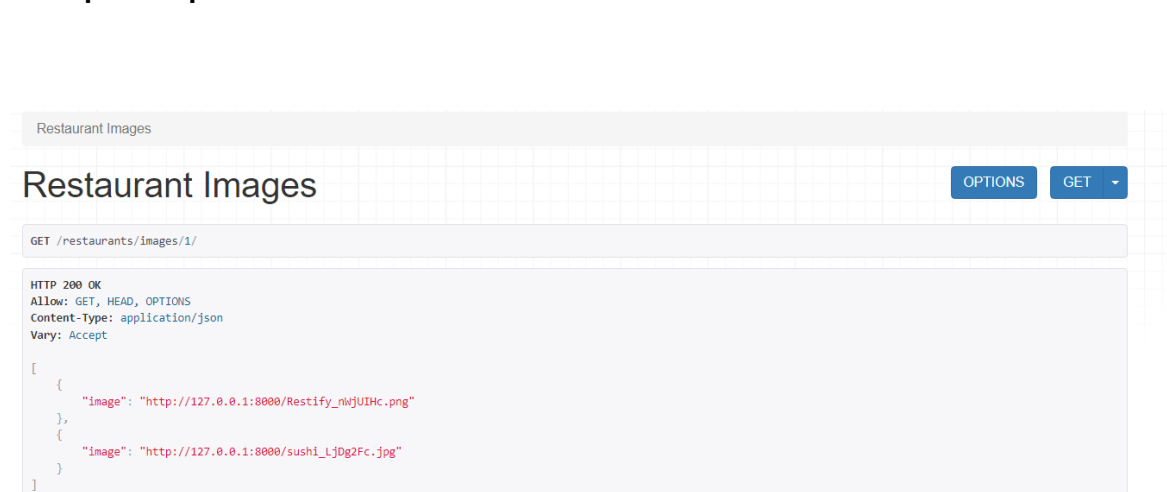
Request

- Url params: <restaurant_id>
- Permission: None
- Request body
 - None

Response:

- 200 OK: Returns the images of the restaurant with <restaurant_id>
- 404 Not Found: The restaurant does not exist

Example Response Success:



Example Response Failure:

Restaurant Images

Restaurant Images

OPTIONS

GET ▾

GET /restaurants/images/785/

HTTP 404 Not Found
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{  
  "detail": "Not found."  
}
```