CSC309

Restify Documentation

Abdul M., Rajath S. Andriy D.

March 15, 2022

Table of Content

Project Overview

- Restaurant
    - Contains everything pertaining to the restaurant aspect of the RESTIFY application
    - Ability to add/edit restaurants, add/remove blog posts, add/update menu items, add/delete restaurant pictures and search through list of restaurants
- Socials
    - Contains everything pertaining to the social aspect of the RESTIFY application
        - Ability to follow/unfollow restaurants, like/dislike restaurants, like/dislike blog posts, view user feed, add and view comments and view user notifications
- User
    - Contains everything pertaining to the user aspect of the RESTIFY application
        - Ability to register users, login, refresh token and update user profile

Project Structure

- Models
  - Models for the restaurant and socials app are located in a models directory, while the users app has models located in models.py
- Views
  - All views for each app are contained within the respective views directory
- Seralizers
  - All the serializers can be found within the respective serializer.py file found in each app
- Shell files
  - Both of these can be found within the P2 directory
    - Startup.sh is run initially to set up our project
      - Creating and activating the Venv
      - Installing the pip requirements file
      - Making all the appropriate migrations
    - Run.sh is run after Startup.sh to actually run the server
      - Venv is sourced again
      - Django server is started

API Documentation

- NOTE:
    - For all endpoints that use an image upload, please use pass in fields via form-data (not json)
    - For all non image fields use the “text” drop down (except menuItem price can be float), for images use the “file” dropdown
    - For adding/deleting anything user needs to be authenticated (using postman bearer token)

Restaurant Endpoints

- Endpoint: /restaurants/add/
    - Methods: POST
    - Fields (passed in through form-data to accommodate image upload): name, address, postalCode, logo, type, description
    - Description: Allows the user to add a restaurant to which they’re the owner. All fields except the logo are text fields. Logo must be an uploaded file
    - Sample Response Body:

    ```
    {"name":"Dominos", "address": "2555 Erin Centre BLVD",
    "logo":"Dominos_pizza_logo.png", "description":"Dominos"}
    ```
    - Additional notes: the owner isn’t required to be specified, set to whoever logged in
- Endpoint: getRestaurant/<id>/

- Methods: GET
- Description: Allows the user to get complete information on a restaurant
- Example Response:

```
{
    "id": 1,
    "name": "McDonalds",
    "address": "123 Street",
    "postalCode": "L5V1N5",
    "phoneNumber": "911",
    "logo": "http://localhost:8000/1_E7ltikDy9rWcALeVWjnSUg.jpeg",
    "description": "DEEEF",
    "type": "IN",
    "owner": 1,
    "menu": [
        {
            "id": 1,
            "name": "efefewf",
            "description": "dfdf",
            "price": "22222.00",
            "type": "RD",
            "restaurant": 1
        },
        {
            "id": 2,
            "name": "Carrot",
            "description": "Orange",
            "price": "999.00",
            "type": "RD",
            "restaurant": 1
        },
          ],
    "likes_count": 0,
    "dislikes_count": 1,
    "followers_count": 1,
    "comments": [         {
            "id": 5,
            "title": "HELLO",
            "content": "THIS IS A COMMENT POST",
            "date": "2022-03-15T15:27:50.943758Z",
            "name": "def",
            "owner": 1,
            "restaurant": 1
        },
        {
            "id": 6,
            "title": "HELLOOOOO",
            "content": "THIS IS A COMMENTPOST",
            "date": "2022-03-15T15:29:11.832448Z",
            "name": "def",
            "owner": 1,
            "restaurant": 1
        },      ],
    "blog_posts": [
        {
            "id": 1,
            "title": "HELLO",
            "short_description": "SHORT DESCRIPTION",
            "content": "HELLLLOOO",
            "date": "2022-03-15T21:52:51.775706Z",
            "restaurant": 1,
            "like_status": "None"
        },
        {
            "id": 2,
            "title": "efefewf",
            "short_description": "dfdf",
            "content": "22222",
            "date": "2022-03-16T02:48:49.357529Z",
            "restaurant": 1,
            "like_status": "None"
        }
    ],
    "follow_status": "None",
    "like_status": "None",
    "images": [
        {
            "id": 1,
            "alt_code": null,
            "image": "/3WjLKcw.gif",
            "restaurant": 1
        }
    ]
}
```

    ○ Additional notes: any user can view restaurant information, but the like and follow status for blogs and restaurant is customized to the user sending the request

- Endpoint: /restaurants/update/
    ○ Methods: PATCH
    ○ Fields (Request Body): name, address, postalCode, logo, type

- Description: Allows the user to edit a restaurant which they own
- Example body:

```
{"name":"Dominos", "address": "2555 Erin Centre BLVD",
"logo":"https://upload.wikimedia.org/wikipedia/commons/thumb/7/
74/Dominos_pizza_logo.svg/1200px-Dominos_pizza_logo.svg.png","d
escription":"Dominos"}
```

- Endpoint: /restaurants/addMenuItem/
  - Methods: POST
  - Fields (Request Body): restaurant, name, description, price, type
  - Description: Allows the user to add a menuItem to a restaurant. 'restaurant' is the ID of the restaurant to which the menu item is being added. Request user MUST own this restaurant
  - Example body:

```
{  "id": 5, "name": "sushi", "description":
"yum2","price": "100213.00", "type":"FD", "restaurant": 1}
```

  - Additional notes: the restaurant id will be passed via the front-end in sprint 3. The user also needs to be authenticated, to add to a specific restaurant.
- Endpoint: /restaurants/updateMenuItem/<id>/
  - Methods: PATCH
  - Fields (Request Body): restaurant, name, description, price, type
  - Description: Allows the user to edit a menu item for their restaurant
  - Example body:

```
{"name":"food"}
```

    - Additional notes: the restaurant id will be passed via the front-end in sprint 3. Whichever fields are not passed in the body will keep their old values
- Endpoint: /restaurants/addBlog/
    - Methods: POST
    - Fields: title, short_description, restaurant (id), content, like_count, dislike_count
    - Description: Allows the user to add a blog post for their restaurant
    - Example body:

```
{ "title" : "Dominos Rocks!", "short_description" :
"cool", "restaurant" : 2, "content" : "Very good food",
"like_count" : 300, "dislike_count" : 1 }
```

    - Additional notes: the restaurant id will be passed via the front-end in sprint 3. Whichever fields are not passed in the body will keep their old values. User must be authenticated.
- Endpoint: /restaurants/deleteBlog/<id>/
    - Methods: DELETE
    - Fields: blogPost id (URL)
    - Description: Allows the user to delete a blog post off their restaurant
    - Example body:

      N/A
    - Additional notes: the blogpost id will be passed via the front-end in sprint 3 via the URL. User must be authenticated.
- Endpoint: /restaurants/getBlogPost/<id>/
    - Methods: GET
    - Fields: blogPost id (URL)
    - Description: Allows the user to get a blog post off their restaurant

    - Example body:

```
{ "id": 16,    "title": "testBlog",
"short_description": "wo2",
"content": "very cool",    "like_count": 3,
"dislike_count": 1,    "restaurant": 2}
```

    - Additional notes: the blogpost id will be passed via the front-end in sprint 3 via the URL. User must be authenticated.
- Endpoint: /restaurants/addPicture/
    - Methods: POST
    - Fields: alt_code, restaurant (id), content, image
    - Description: Allows the user to add a picture for their restaurant
    - Example body:

```
{ "alt_code" : "Nice picture", "restaurant" : 2, "image":
thaiexpress.jpg}
```

    - Additional notes: the restaurant id will be passed via the front-end in sprint 3. Alt_code is not required. User must be authenticated. For the image, form data with image dropdown should be selected/passed
- Endpoint: /restaurants/deletePicture/<id>/
    - Methods: DELETE
    - Fields: picture id (URL)
    - Description: Allows the user to delete a picture off their restaurant
    - Example body:

      N/A

    - Additional notes: the picture id will be passed via the front-end in sprint 3 via the URL. User must be authenticated.
- Endpoint: /restaurants/list/?keyword=<insertSearchTerm>

- Methods: GET
- Fields: Keyword (search term passed via URL)
- Description: Allows the user to delete a picture off their restaurant

Socials Endpoints

- Endpoint: /socials/<restaurant_id>/like_or_dislike/
    - Methods: POST
    - Fields: type (‘Like’ or ‘Dislike’)
    - Description: Allows the user to like or dislike a restaurant
    - Example body:

    ```
    {type: "Like"}
    ```
    - Additional notes: User must be authenticated to “react” to a restaurant
- Endpoint: /socials/<restaurant_id>/update_like_or_dislike/
    - Methods: PATCH
    - Fields: type (‘Like’ or ’Dislike’)
    - Description: Allows the user to update their like/dislike choice for a restaurant
    - Example body:

    ```
    {type: "Like"}
    ```
    - Additional notes: User must be authenticated to update their reaction to a restaurant
- Endpoint: /socials/<restaurant_id>/delete_like_or_dislike/
    - Methods: DELETE

    - Description: Allows the user to delete their like/dislike choice for a restaurant
    - Additional notes: User must be authenticated to delete their reaction to a restaurant
- Endpoint: /socials/<restaurant_id>/follow/
    - Methods: POST
    - Fields: N/A
    - Description: Allows the user to follow a restaurant
    - Additional notes: User must be authenticated to follow a restaurant
- Endpoint: /socials/<restaurant_id>/unfollow/
    - Methods: DELETE
    - Fields: N/A
    - Description: Allows the user to unfollow a restaurant
    - Additional notes: User must be authenticated to unfollow a restaurant
- Endpoint: /socials/feed/
    - Methods: GET
    - Fields: N/A
    - Description: Allows the user to get a feed of blog posts from restaurants that they follow
    - Additional notes: User must be authenticated to view their feed
- Endpoint: /socials/blog/<blog_id>/like_or_dislike/
    - Methods: POST
    - Fields: type (‘Like’ or ’Dislike’)
    - Description: Allows the user to like/dislike a restaurant’s blog post
    - Example body:

```
{type: "Like"}
```

  - Additional notes: User must be authenticated to like/dislike a restaurant's blog posts
- Endpoint: /socials/blog/<blog_id>/update_like_or_dislike/
  - Methods: PATCH
  - Fields: type ('Like' or 'Dislike')
  - Description: Allows the user to update their reaction of a blog post
  - Example body:

```
{type: "Like"}
```

  - Additional notes: User must be authenticated to update their reaction of a blog post
- Endpoint: /socials/blog/<blog_id>/delete_like_or_dislike/
  - Methods: DELETE
  - Fields: N/A
  - Description: Allows the user to delete their reaction of a blog post
  - Additional notes: User must be authenticated to delete their reaction of a blog post
- Endpoint: /socials/<restaurant_id>/comment/
  - Methods: POST
  - Fields: title, content
  - Description: Allows the user to comment on a restaurant
  - Example body:

```
{title: "POG", content: "Not POG"}
```

    - Additional notes: User must be authenticated to comment on a restaurant

- Endpoint: /socials/comments/list/
    - Methods: GET
    - Fields: N/A
    - Description: Allows an individual to get all comments
    - Additional notes: User must be authenticated to get all comments.
- Endpoint: /socials/notifs/
    - Methods: GET
    - Fields: N/A
    - Description: Allows an individual to get all their notifications
    - Additional notes: User must be authenticated to get all notifications.

User Endpoints

- Endpoint: /user/register/
    - Methods: POST
    - Fields (pass in through form data to allow image upload): email, password, first_name, last_name, avatar (image), phone_number
    - Description: Allows a new user to be registered
- Endpoint: /user/login/
    - Methods: POST
    - Fields: email, password
    - Description: Allows a user to login and retrieve access and refresh tokens
- Endpoint: /user/update/

    - Methods: PUT
    - Fields (pass in through form data to allow image upload): password, first_name, last_name, avatar (image), phone_number
    - Description: Allows the logged in user to update their profile. Users cannot update their email as the email also functions as username
- Endpoint: /user/refresh/
    - Methods: POST
    - Fields: refresh
    - Description: Allows you to refresh access token by passing in a refresh token