Restify

Setup / Running the Server

(May need to chmod 777 the startup.sh, run.sh, and manage.py files however last we tested it should be working) Assumptions are that the VM has python3.10 under python3 and pip3 and pip already installed.

Please run the startup.sh file under the command

source startup.sh

Accessing APIs

Note: You can ignore this if you are using Visual Studio as it should handle port forwarding automatically.

Run Server Potential Issues

You will have to change the run.sh file to include the ip address of the VM in some cases of port forwarding. You will need to edit the following line in run.sh from:

```
python3 manage.py runserver
```
to:
```
python3 manage.py runserver 192.168.xx.xxx:8000
```

(192.168.xxx.xxx being the IP address of your VM found by ifconfig)

Now you should be able to access your server on your host machine's browser at 192.168.xx.xxx:8000.

This will most likely cause an issue with django and you will get an error telling you to add the IP to ALLOWED_HOSTS.

cd into RESTify/ and edit in settings.py add '192.168.xx.xxx' to the ALLOWED_HOSTS list.

List of APP’s

Our backend consists of three apps: “**accounts**”, “**restaurants**” and “**blogs**”. Accounts takes care of all the api’s related to the user such as signup, login and profile. Restaurants handle anything (except blogs) that has to do with having a restaurant or interacting with a restaurant like liking, following and notifications. Blogs take care of adding blogs and liking blogss.

Model Structure

accounts

- **Abstract User -** This model just add new fields to the existing Django User Model
  - An phone field is for phone numbers which uses a PhoneNumberField from https://github.com/stefanfoulis/django-phonenumber-field
  - An avatar field is added for profile picture and by default avatar.png is every users profile picture which they can change later on

restaurants

- **Restaurant** - This model store each restaurant and has it connected to a owner
  - name - restaurants name
  - address - address of restaurant
  - postal - postal code of restaurants location
  - phone - phone number of restaurant
  - logo - picture of logo of restaurant
  - description - restaurant description
  - cover_img - restaurant's cover image
  - carousel_img_1 - first image for carousel
  - carousel_img_2 - second image for carousel
  - carousel_img_3 - third image for carousel
  - image_1 - additional image on the restaurant page
  - image_2 - additional image on the restaurant page
  - image_3 - additional image on the restaurant page
  - image_4 - additional image on the restaurant page

- **RestaurantNotifications -** This model is for keeping track of all the notifications that users who follow the restaurant can see.
  - restaurant - Restaurant Model
  - title - the notification message
  - last_modified - the time the notification was added

- **OwnerNotifications -** This model keeps track of all the notifications a owner of a restaurant gets when other users interact with their restaurant.
  - restaurant - Restaurant Model
  - user - Abstract User Model
  - title - the notification message
  - last_modified - the time the notification was added

- **Following -** This model keeps track of which user follows which restaurant
  - user - Abstract User Model
  - restaurant - Restaurant Model

- **Comment -** This model keep track of the comment user writes on a restaurant
    - user - Abstract User Model
    - restaurant - Restaurant Model
    - comment - the actual comment text
    - datetime - time when comment was added

- **RestaurantLike -** This model keeps track of which users like which restaurant.
    - user - Abstract User Model
    - restaurant - Restaurant Model

- **MenuItem -** This model keeps track of every single menu item entry for each restaurant. Note restaurants can have many menu items
    - restaurant - Restaurant Model
    - name - name of the menu item
    - description - menu item description
    - price - menu item price
    - type - type of menu item (Appetizers, Main Course, Sides, Specials, Desserts, Drinks)

blogs

- **BlogPost -** This model keeps track of all the blog posts of all the restaurants.
    - user - Abstract User Model
    - restaurant - Restaurant Model
    - title - title of blog post
    - description - blog post in text
    - primary_photo - blog posts main photo
    - photo_1 - additional photo to be displayed
    - photo_2 - additional photo to be displayed
    - photo_3 - additional photo to be displayed
- **BlogLike -** This model keeps track of all the likes of all the blog posts by users.
    - user - Abstract User Model
    - blog_post - Blog Post Model

Accounts [APP - Accounts]

AUTHENTICATED	HTTP Method	Payloads	Error	Successful Result
/accounts/signup/ POST: Creates a user profile				
▨	**POST**	username: unique username **(required)** first_name: first name last_name: last name email: valid email phone: phone number in E.164 format password: at least 8 characters **(required)** password2: same as password1 **(required)**	**400 - Bad Request:** username: not valid or unique password: too short or does not match phone: invalid format (E.164) email: not valid email	**201 - CREATED:** username first_name phone last_name email
/accounts/login/ **/accounts/api/token/** POST: Provides the access token when user logs in				
▨	**POST**	username: unique username **(required)** password: user password **(required)**	**400 - Bad Request:** username: not valid password: not valid	**200 - OK:** refresh access
/accounts/profile/ **/accounts/profile/edit/** GET: Lists user profile information				
AUTHENTICATED	**GET**	▨	**401 - Unauthorized:** user not logged in	**200 - OK:** username first_name last_name email phone avatar

/accounts/profile/edit/
/accounts/profile/

PUT: Edit the user profile
Note: to change user the password, then old_password, new_password, new_password2 all need to be filled out with sign-up rules in mind (eg password has to be longer than 8 character and new_password and new_password2 have to match
Note: the format of phone has to be in E.164 format to be recognized as a valid phone number.

AUTHENTICATED	**PUT**	first_name: update first name last_name: updated last name email: updated valid email avatar: updated image old_password: old password new_password: new password new_password2: confirm new password	**401 - Unauthorized:** user not logged in **400 - Bad Request:** password: to short or does not match phone: invalid format (E.164) email: not valid email	username first_name last_name phone email avatar

Following [APP - Restaurants]

AUTHENTICATED	HTTP Method	Payloads	Error	Successful Result
/restaurants/follow/rest_id/ Get: user can find out if they follow restaurant or not (true or false returned) POST: user can start following restaurant DELETE: user can stop following restaurant *NOTE: if a user attempts to follow a restaurant it is already following then this will result in 200 OK and same with deleting a follow (unfollow).*				
AUTHENTICATED	**GET**	<int:rest_id>: id of restaurant	**401 - Unauthorized:** user not logged in **404 - Not Found:** rest_id: invalid restaurant id	**200 - OK:** result: {boolean}
	POST	<int:rest_id>: id of restaurant	**401 - Unauthorized:** **user not logged in** **404 - Not Found:** rest_id: invalid restaurant id	**200 - OK:** Success: User already follows the restaurant. **OR** Success: User now follows the restaurant.
	DELETE	<int:rest_id>: id of restaurant	**401 - Unauthorized:** user not logged in **404 - Not Found:** rest_id: invalid restaurant id	**200 - OK:** Success: User never followed the restaurant. **OR** Success: User does not follow the restaurant anymore.

Restaurant Notifications [APP - Restaurants]

AUTHENTICATED	HTTP Method	Payloads	Error	Successful Result
/restaurants/notification/all/ GET: returns total number of notification received, url for next set of notifications, url for previous set of notifications and a list of all notification ordered by add time for all the restaurants the user follows *Note: result is an ordered list of notifications with restaurant id, notification title, time notification was uploaded*				
AUTHENTICATED	GET		**401 - Unauthorized:** user not logged in	**200 - OK:** count next previous result: [[restaurant, title, last_modified], ……….]
/restaurants/notification/rest_id/ GET: returns total number of notification received, url for next set of notifications, url for previous set of notifications and a list of all notification ordered by add time (format of list: restaurant id, title, time notification was added) associated with rest_id *Note: result is an ordered list of notifications with restaurant id, notification title, time notification was uploaded*				
AUTHENTICATED	GET	<int:rest_id>: id of restaurant	**401 - Unauthorized:** user not logged in **404 - Not Found:** rest_id: invalid restaurant id	**200 - OK:** count - number of notification next - url for next set previous - url for previous set result: [[restaurant_id, title, last_modified], ……….]
/restaurants/notification/blog/rest_id/ **/restaurants/**notification/menu/rest_id/ POST: add notification about the restaurant like when the owner either updates their restaurant menu or adds a blog.				
AUTHENTICATED	POST	<int:rest_id>: id of restaurant	**401 - Unauthorized:** **403 - Forbidden:** rest_id: is not owned by user **404 - Not Found:** rest_id: invalid restaurant id	**200 - OK:** id - notification id restaurant - restaurant name title - notification message

Owner Notifications [APP - Restaurants]

Address /restaurants/…	HTTP Method	Payloads	Error	Successful Result
/restaurants/owner/update/follow/rest_id/ **/restaurants/owner/update/comment/rest_id/** **/restaurants/owner/update/like/rest/rest_id/** **/restaurants/owner/update/like/blog/rest_id/** POST: Add notification for when a user reacts to a Restaurant by a follow, liking restaurant or blog, or commenting				
AUTHENTICATED	**POST**	<int:rest_id>: id of restaurant	**400 - Bad Request:** rest_id: invalid restaurant id	**200 - OK:** id - notification id title - notification message restaurant - restaurant name
/restaurants/owner/update/all/ GET: returns total number of notification received, url for next set of notifications, url for previous set of notifications and a list of all notification ordered by add time which have to do with all the interactions other users have had with logged in users restaurant				
AUTHENTICATED	**GET**		**401 - Unauthorized:** **user not logged in**	**200 - OK:** count next previous result: [[restaurant, title, last_modified], ……….]

Restaurant Likes [APP - Restaurants]

AUTHENTICATED	HTTP Method	Payloads	Error	Successful Result
/restaurants/restaurant_id/like/add/ Creates a RestaurantLike model object (many to one relationship to a Restaurant). Lets users like a restaurant.				
AUTHENTICATED	POST	▨	**400 - Bad Request:** user trying to like their own restaurant **404 - Not Found** no restaurant associated with restaurant_id in url **401 - Unauthorized** invalid or missing bearer token	**201 - CREATED:** Success: User already likes the restaurant. **OR** Success: User now likes the restaurant.
/restaurants/restaurant_id/like/ GET: used to determine whether user has liked the restaurant with restaurant_id (True or 404) DELETE: removes a RestaurantLike model object. Allows user to remove their "like" from a restaurant				
AUTHENTICATED	GET	<int:restaurant_id>: id of restaurant	**404 - Not Found** user has not liked the restaurant associated with restaurant_id **OR** no restaurant associated with restaurant_id in url **401 - Unauthorized** invalid or missing bearer token	**200 - OK** result: boolean
	DELETE	<int:restaurant_id>: id of restaurant	**401 - Unauthorized** invalid or missing bearer token **404 - Not Found** user has not liked the restaurant associated with restaurant_id **OR** no restaurant associated with restaurant_id in url	**204 - NO CONTENT**

Restaurant [APP - Restaurants]

AUTHENTICATED	HTTP Method	Payloads **(Bold parameters are required)**	Error	Successful Result
/restaurants/add/ Creates a restaurant that belongs to the requesting user				
AUTHENTICATED	**POST**	name: restaurant name **(required)** address: restaurant address **(required)** postal: postal of restaurant address **(required)** phone: restaurant phone number in E.164 format **(required)** logo: restaurant logo **(required)**	**400 - Bad Request:** missing/blank field: name, address, postal, phone, logo **OR** phone number is invalid **OR** postal code is invalid **OR** user already owns a restaurant **401 - Unauthorized** invalid or missing bearer token	**201 - CREATED:** name address postal phone logo num_likes: 0
/restaurants/search/ Retrieve a JSON array of restaurants filtered by the provided search parameter value (search query filters by food, name and / or address), ordered by num_likes descending. If no search parameter is specified or the search parameter is an empty string, return no results.				
	GET	(search field must be In Params, not Body) Key: search Value: search query		**200 - OK** count - number of restaurants next - url for next set previous - url for previous set results: [{name address postal phone logo num_likes }, …]

/restaurants/restaurant_id/edit/ Edit the restaurant values associated with the requesting user. If no value is provided, the original value will be assumed. Only description, cover_img, carousel_img_# and image_# fields can be left blank to remove the image/textfield.				
AUTHENTICATED	**PUT**	name: restaurant name address: restaurant address postal: postal of restaurant address phone: restaurant phone number in E.164 format logo: restaurant logo description: restaurant page info cover_img: restaurant page image carousel_img_1: restaurant page image carousel_img_2: restaurant page image carousel_img_3: restaurant page image image_1: restaurant page image image_2: restaurant page image image_3: restaurant page image image_4: restaurant page image <int:restaurant_id>: id of restaurant	**400 - Bad Request:** invalid postal invalid phone invalid image **403 - Forbidden** user trying to edit another user's restaurant **401 - Unauthorized** invalid or missing bearer token **404 - Not Found** restaurant_id in url is not associated to any restaurant	**200 - OK** name address postal phone logo description cover_img carousel_img_1 carousel_img_2 carousel_img_3 image_1 image_2 image_3 image_4
/restaurants/restaurant_id/view/ Return the information corresponding to the provided restaurant_id				
	GET	<int:restaurant_id>: id of restaurant	**404 - Not Found** restaurant_id in url is not associated to any restaurant	**200 - OK** name address postal phone logo description cover_img carousel_img_1 carousel_img_2 carousel_img_3 image_1 image_2 image_3 image_4 num_likes num_follows num_comments num_blogs

Restaurant Menu [APP - Restaurants]

AUTHENTICATED	HTTP Method	Payloads (Bold parameters are required)	Error	Successful Result
restaurants/restaurant_id/menu/ View all menu items for a specific restaurant id				
▨	**GET**	<int:restaurant_id>: id of restaurant	**404 - Not Found:** invalid restaurant_id restaurant does not exist	**200 - OK:** count - number of menu items next - url for next set previous - url for previous set result: [[name, description, price, type], ……….]
restaurants/menu_item_id/editMenuItem/ Edit a specific menu item using menu item id. If certain Payloads is not provided keeps as old data				
AUTHENTICATED	**PUT**	name: item name description: item description price: item price type: item type <int:menu_item_id>: id of menu item	**404 - Not Found** invalid menu_item_id menu item does not exist **400 - Bad Request:** Invalid price, was below 0	**200 - OK** name description price type
restaurants/addMenuItem/ Create a menu item for the signed in user (must have / be the owner of a restaurant)				
AUTHENTICATED	**POST**	name: item name **(required)** description: item description **(required)** price: item price **(required)** type: item type **(required)**	**400 - Bad Request:** user trying to add a menu but they don't own a restaurant invalid price **401 - Unauthorized** invalid or missing bearer token	**201 - CREATED** name description price type

Restaurant Comments [APP - Restaurants]

AUTHENTICATED	HTTP Method	Payloads (Bold parameters are required)	Error	Successful Result
restaurants/restaurant_id/viewComments/ View all comments for a specific restaurant id				
	GET	<int:restaurant_id>: id of restaurant	**404 - Not Found** restaurant id does not exist	**200 - OK:** count - number of comments next - url for next set previous - url for previous set result: [[comment, datetime], ……….]
restaurants/restaurant_id/addComment/ Add a comment to a specific restaurant id				
AUTHENTICATED	**POST**	comment: comment for restaurant **(required)** <int:restaurant_id>: id of restaurant	**404 - Not Found** restaurant id does not exist	**201 - CREATED:** comment datetime

Blog Posts [APP - Blogs]

AUTHENTICATED	HTTP Method	Payloads (Bold parameters are required)	Error	Successful Result
blogs/blog_post_id/ Retrieve the blog post with blog_post_id				
▨	**GET**	<int:blogpost_id>: id of blog post	**404 - Not Found:** blog_post_id does not exist	**200 - OK:** restaurant title description primary_photo photo_1 photo_2 photo_3 last_modified
blogs/create/ Create a blog post for the currently signed in users restaurant (only if they own a restaurant)				
AUTHENTICATED	**POST**	title: blog post title **(required)** description: blog post summary **(required)** primary_photo: blog post main photo photo_1: blog post 1st extra photo photo_2: blog post 2nd extra photo photo_3: blog post 3rd extra photo	**400 - Bad Request:** user trying to add a menu but they don't own a restaurant	**201 - CREATED** title description primary_photo photo_1 photo_2 photo_3 last_modified
blogs/blog_post_id/edit/ Edit a blog post according to blog post id that the signed in user owns				
AUTHENTICATED	**PUT**	title: blog post title description: blog post summary primary_photo: blog post main photo photo_1: blog post 1st extra photo photo_2: blog post 2nd extra photo photo_3: blog post 3rd extra photo <int:blog_post_id>: id of blog post	**400 - Bad Request:** user trying to edit a restaurants blog post they do not own **401 - Unauthorized** invalid or missing bearer token **404 - Not Found** blog_post_id in url is not associated to any blog post	**200 - OK** title description primary_photo photo_1 photo_2 photo_3 last_modified

blogs/home/ View all blog posts in order of last modified, according to the restaurants a user follows				
AUTHENTICATED	**GET**		**401 - Unauthorized** invalid or missing bearer token	**200 - OK** count - number of blog posts next - url for next set previous - url for previous set result: [[title, description, primary_photo, photo_1,, photo_2, photo_3, last_modified], ……….]

Blog Post Likes [APP - Blogs]

AUTHENTICATED	HTTP Method	Parameter	Error	Successful Result
/blogs/blogpost_id/like/add/ Creates a BlogLike model object (many to one relationship to a Blog Post). Lets users like a blog post.				
AUTHENTICATED	**POST**	<int:blogpost_id>: id of blog post	**400 - Bad Request:** user trying to like their own blog post **404 - Not Found** no blog post associated with blogpost_id in url **401 - Unauthorized** invalid or missing bearer token	**201 - CREATED:** Success: User already likes the blog post. **OR** Success: User now likes the blog post.
/blogs/blogpost_id/like/ GET: used to determine whether user has liked the blog post with blogpost_id (True or 404) DELETE: removes a BlogLike model object. Allows user to remove their “like” from a blog post.				
AUTHENTICATED	**GET**	<int:blogpost_id>: id of blog post	**404 - Not Found** user has not liked the blog post associated with blogpost_id **OR** no blog post associated with blogpost_id in url **401 - Unauthorized** invalid or missing bearer token	**200 - OK** result: boolean
	DELETE	<int:blogpost_id>: id of blog post	**401 - Unauthorized** invalid or missing bearer token **404 - Not Found** user has not liked the blog post associated with blogpost_id **OR** no blog post is associated with blogpost_id	**204 - NO CONTENT**