General Design decisions:

Likes and follows are written as ManyToManyField, thus when you check in db, There is no ‘following’ column or ‘likes’ column for User. However, rest assured that you can both receive all notifications from those you followed, and view all their blogs (feed link)

For all endpoints, you need to log in first.

For all editing endpoints, you can either post or patch. Patch requires only fields you wish to change, while put will change everything and set null to those not provided. All editing endpoints require owner status to change corresponding content, I implemented custom permission classes in their views.

Pagination: default page size is set to 5 in P2/settings.py. If you want custom page size, just include key=limit, value=[new_size] in Postman Params, example resulting url below:
http://localhost:8000/restaurant/search/?limit=3

Search: for search restaurant, use localhost:8000/restaurant/serach/ endpoint. Include payload [food]=[example_food_name] [name]=[example_rest_name] [address]=[example_addr] in Postman Params, example usage below:

http://localhost:8000/restaurant/search/?limit=3&name=Mcdonald&address=123 Street&food=mcflurry

For all delete endpoints, they only accept DELETE requests. All other requests will result in errors.

We implemented CreateAPIView, RetrieveAPIView, UpdateAIPView, ListAPIView, DeleteAPIView, etc. Pagination is done in ListAPIView.

Auth token is set to expire in 365 days for TA testing convenience.

APPS:

1. accounts: User(AbstractUser)
    a. User: endpoint:
        i. First_name: CharField
        ii. last_name, CharField
        iii. email, EmailField
        iv. avatar, ImageField
        v. phone_number, Charfield
        vi. Phone_regex, RegexValidator,

2. restaurant: restaurant, blog, blog_comment, restaurant_comment
    a. restaurant:
        i. name, CharField
        ii. owner,OneToOneField
        iii. description,CharField
        iv. address,CharField
        v. postal_code,CharField
        vi. followers,ManyToManyField
        vii. Likes, ManyToManyField
        viii. logo,ImageField
        ix. Phone_regex,RegexValidator
        x. phone_number,CharField
    b. Restaurant_comment:
        i. Poster: User
        ii. restaurant:restaurant,
        iii. Content,CharField
        iv. time:datetime,DateTimeField
    c. food:
        i. name, CharField
        ii. restaurant,:Restaurant
        iii. description, CharField
        iv. price, positiveIntegerField
        v. images,ImageField

3. Notification
    a. Notification :
        i. receiver:User
        ii. initiator:User
        iii. Url, URLField
        iv. time.DateTimeField
        v. is_viewed,BooleanField(True)
    b. RestNotification:
        i. receiver:User
        ii. initiator:User
        iii. Url, URLField
        iv. time.DateTimeField
        v. is_viewed,BooleanField(False)
4. Blog()
    a. Blog:
        i. restaurant:restaurant,
        ii. Title,CharField
        iii. image, ImageField
        iv. Content,CharField
        v. likes, ManyToManyField

vi. time,DateTimeField

b. Blog_Comment:
   i. poster: User,
   ii. blog:Blog,
   iii. time, DateTimeField
   iv. content,CharField

APIs:

Endpoint: /accounts/profile/<int:id>/edit/
Description: Updating user's profile whose id equal to to id.
Method:PATCH, PUT
Payload:email, avatar, phone_number, password, password2, first_name, last_name, username,Phone_number

Endpoint:/accounts/profile/register
Description: register a user
Method:GET, POST,HEAD, OPTIONS
Payload:email, avatar, phone_number, password, password2, first_name, last_name, username,phone_number

Endpoint:/restaurant/add/
Description: Add a restaurant of current user. No user can create 2 or more restaurant.
Method: POST
Payload:name, owner, description,address,postal_code,followers, likes, logo,phone_number

Endpoint:/restaurant/search/
Description: search restaurant with the keyword we put in payload.
Method: POST
Payload:food,name,address

Endpoint:/restaurant/<int:rest_id>/
Description: Show the information of the Restaurant with id equal to rest_id.
Method:GET
Payload: None

Endpoint:/restaurant/<int:rest_id>/edit/
Description:Editing Restaurant. Only owner of rest is allowed to edit.
Method:PATCH, PUT
Payload:name, owner, description,address,postal_code,followers, likes, logo,phone_number

Endpoint:/restaurant/<int:rest_id>/follow/

Description: Following and unfollowing restaurant with id equal to rest_id. If current user is not the followers of the restaurant, post to become follower of this restaurant. If not, POST to unfollow the restaurant.

Method:POST

Payload:name, owner, description,address,postal_code,followers, likes, logo,phone_number

Endpoint:/restaurant/<int:rest_id>/like/

Description: Liking the restaurant with id equal to rest_ud. If current user is not in the like list of the restaurant, Post to like this restaurant. If not, POST to unlike the restaurant.

Method:POST

Payloadname, owner, description,address,postal_code,followers, likes, logo,phone_number

Endpoint:/restaurant/<int:rest_id>/menu/

Description: view the menu of restaurant with id equals to rest_id.

Method:GET

Endpoint:/restaurant/<int:rest_id>/menu/add/

Description: 1. adding the menu, only owner allowed 2. Send notification to the followers.

Method:POST

Payload:name, owner, description,address,postal_code,followers, likes, logo,phone_number

Endpoint:/restaurant/menu/<int:food_id>/delete/

Description: delete the food, only owner allowed

Method:DELETE

Endpoint:/restaurant/<int:rest_id>/menu/<int:food_id>/edit/

Description: 1.update the food view 2. Send notification to the followers of the restaurant.

Method:PATCH, PUT

Payload:name, restaurant, price, description, images

Endpoint:/restaurant/<int:rest_id>/comments/

Description: view comments of the restaurant with id equals to rest_id

Method:GET

Endpoint:/restaurant/<int:rest_id>/comments/add/

Description:1. Current user add comment to the restaurant. 2. Send notification to the owner of the restaurant.

Method: POST

Payload:poster, restaurant, content, time

Endpoint:/restaurant/<int:rest_id>/comments/<int:rest_comm_id>/edit/

Description: If the current user is the poster of the comment, current user can edit the comment.

Method:PATCH, PUT

Payload:poster, restaurant, content, time

Endpoint:/restaurant/<int:rest_id>/comments/<int:rest_comm_id>/delete/

Description: If the current user is the poster of the comment(id = rest_comm_id) or the owner of the restaurant with id = rest_id, the current user can delete the comment.

Method:DELETE

Endpoint:/blog/feed/

Description: the current user can view the feed page of blogs.

Method:GET

Endpoint:/blog/add/

Description: 1. The owner can add blog post of its own restaurant. 2. It will send notification to the followers of this restaurant.

Method:GET, POST

Payload:title, content, images, time, likes

Endpoint:blog/<int:blog_id>/

Description: view the blog with id = blog_id

Method:GET

Endpoint:blog/<int:blog_id>/edit/

Description:Only if the current user is the poster of the blog which id equals to blog_id, the user can edit this blog.

Method:PATCH, PUT

Payload:title, content, images, time, likes

Endpoint:blog/<int:blog_id>/delete/

Description:Only current user is the poster of the blog which id =blog_id , user can delete this blog.

Method:DELETE

Endpoint:blog/<int:blog_id>/like

Description: 1.If current user not like the blog post with id = blog_id, Post will add Current user in the like list. It will send the notification to the blog poster. 2. If current user already liked the post, Post will remove the current user from the like list of this blog post.

Method:POST

Endpoint:blog/<int:blog_id>/comments/add/

Description: 1. Current user add a blog comment of the blog with id equal to blog_id. 2. What's more, it will create a notification to the owner who posts the blog.

Method: POST

Payload:poster,blog,time,content

Endpoint:blog/<int:blog_id>/comments/

Description: view all blog comments of the blog post with id equals to blog_id

Method:GET

Endpoint:blog/<int:blog_id>/comments/<int:blog_comm_id>/

Description: view the comment with id equal to blog_comm_id.

Method:GET

Endpoint:blog/<int:blog_id>/comments/<int:blog_comm_id>/edit/

Description: edit a blog with id equals to blog_comm_id. only owner allowed

Method:PATCH, PUT

Payload:poster,blog,time,content

Endpoint: blog/<int:blog_id>/comments/<int:blog_comm_id>/delete/

Description: delete a blog comment with id equals to blog_comm_id. only owner allowed

Method:DELETE

Endpoint:notification/view/

Description: The current user view all notification which are not be viewed. And these notifications will be set to viewed.

Method:GET