MODELS

User

Description: The User model extends Django's default user view, and is used to store EasyChef users.

Fields:

- username (CharField)
- password (CharField)
- email (EmailField)
- first_name (CharField)
- last_name (CharField)
- phone_num (CharField)
- avatar (ImageField)

Foreign Keys:

- RecipeModel for recipes the user owns (related name: recipes)
- InteractionModel for that user's interactions (related name: interactions)
- ShoppingRecipeModel for recipes in the user's shopping cart (related name: shoppingCartItems)

ShoppingRecipeModel

Description: The ShoppingRecipeModel holds the information for one recipe in a user's shopping cart.

Fields:

- user_id (ForeignKey REFERENCES User)
- recipe_id (ForeignKey REFERENCES RecipeModel)
- servings_num (PositiveIntegerField)

RecipeModel

Description: The RecipeModel contains all the basic information for a recipe. Steps, Interactions, and Media have their respective foreign keys, as they have a many-to-one relationship with RecipeModel.

Fields:

- user_id (ForeignKey REFERENCES User)
- based_on (ForeignKey REFERENCES self)
- name (CharField)
- total_reviews (IntegerField)
- total_likes (IntegerField)
- total_favs (IntegerField)
- avg_rating (FloatField)
- published_time (DateTimeField)
- difficulty (IntegerField)
- meal (IntegerField)
- diet (IntegerField)
- cuisine (IntegerField)
- total_time (DurationField)
- cooking_time (DurationField)
- prep_time (DurationField)
- calculated_total_time (DurationField)
- calculated_cook_time (DurationField)
- calculated_prep_time (DurationField)
- servings_num (IntegerField)

Foreign Keys:

- StepModel for that recipe's steps (related name: steps)
- InteractionModel for interactions concerning that recipe (related name: interactions)
- RecipeMediaModel for that recipe's uploaded media (related name: media)

RecipeMediaModel

Description: The RecipeMediaModel holds an image associated with an RecipeMedia object. These images are the ones displayed on the banner/carousel of the Recipe, and NOT the step media.

Fields:

- recipe_id (ForeignKey REFERENCES RecipeModel)
- media (FileField)

IngredientModel

Description: The IngredientModel holds all ingredients for all recipes.
Fields:

- Recipe_id (ForeignKey REFERENCES RecipeModel)
- Name (CharField)
- Quantity (IntegerField)
- Unit (CharField default=”cups”)

Foreign Keys:

- Recipe_id which references the recipe this ingredient is for.

StepModel

Description: The StepModel holds all the steps for every recipe.
Fields:

- Recipe_id (ForeignKey REFERENCES RecipeModel)
- Step_num (PositiveIntegerField default=0)
- Cooking_time (DurationField)
- Prep_time (DurationField)
- Instructions (CharField)

Foreign Keys:

- Recipe_id which references the recipe this step is for.

StepMediaModel

Description: The StepMediaModel holds a media file for a specific step in a recipe.
Fields:

- Step_id (ForeignKey REFERENCES StepModel default id)
- Media (FileField) displays one image or video

Foreign Keys:

- Step_id references the step this media file is for.

Additional Notes:

- We had to create StepMediaModel as an intermediate model to store the media files for a step because each step can have multiple media files. We tried looking into an arrayfield to create a field in the StepModel that can store all media files for a step, but arrayfield is specific to postgresql.

InteractionModel

Description: The InteractionModel stores information regarding an interaction (like, favourite, comment) between the user at user_id and the recipe at recipe_id. If a review exists, the media for that review have their own foreign key, as there can be multiple media for each review.

Fields:

- recipe_id (ForeignKey REFERENCES RecipeModel)
- user_id (ForeignKey REFERENCES User)
- like (BooleanField)
- favourite (BooleanField)
- rating (PositiveIntegerField)
- comment (CharField)
- published_time (DateTimeField)

Foreign Keys:

- ReviewMediaModel for any media attached to that interaction's review (related name: media)

ReviewMediaModel

Description: The ReviewMediaModel holds an image associated with an InteractionModel object.

Fields:

- interaction_id (ForeignKey REFERENCES InteractionModel)
- media (FileField)

API ENDPOINTS

User Registration

Description: The Signup View allows the creation of a new user.
Endpoint: http://127.0.0.1:8000/accounts/signup/
Methods: POST
Fields/Payload: username, password, password2, email, first_name, last_name, phone_num
Additional Notes:

- username, password, and password2 fields are required
- The username field must be unique
- The password1 and password2 fields must match
- If the email field is provided, it must contain a valid email
- The avatar of the created user will be a default avatar, found at 'uploads\avatars\default.png'

User Login

Description: The Login View allows users to log into Easy Chef.
Endpoint: http://127.0.0.1:8000/accounts/login/
Methods: POST
Fields/Payload: username, password
Additional Notes:

- username and password fields are required

User Logout

Description: The Logout view allows users to escape Easy Chef.
Endpoint: http://127.0.0.1:8000/accounts/logout/
Methods: GET
Fields/Payload: None

Edit User Profile

Description: The Edit User Profile view allows users to change their password, email, first_name, last_name, phone_num, and avatar.
Endpoint: http://127.0.0.1:8000/accounts/profile/edit/
Methods: PATCH
Fields/Payload: password, password2, email, first_name, last_name, phone_num, avatar
Additional Notes:

- Only the currently authenticated (logged in) user's information will be changed
- If any fields are left blank in the request, that field will not be updated in the User
- If password is provided, then password and password2 must match
- If email is provided, it must be a valid email
- If avatar is provided, it must be a file upload containing an image file

Creating A Recipe:

We assume when a user creates a recipe there already exists a predefined list of ingredients that the user can select and customize based on quantity and unit.

Before sending the POST request to create a recipe, you must send requests for the following:

1. Add Recipe Media (add one or more images of the overall recipe)
2. Add Steps (add one or more steps)
3. Add Step Media (add one or more media to 1 step)
4. Add Ingredients (to add more base ingredients)

1) Add Recipe Media

Description: The Add Recipe endpoint allows users to add one or more images of their recipe they want to create. This feature is restricted to authorized users.

Endpoint: http://127.0.0.1:8000/recipes/create-recipes/add-media/

Methods: POST

Fields/Payload: media

Example Response:
{ "id": 2, "recipe_id": null, "media": "http://127.0.0.1:8000/media/recipe-media/pic.png" }

Additional Notes:

- Upon success, the response above will be sent.
- recipe_id is initially set to null and then will be updated when added to the Create a Recipe endpoint

2) Add Step(s)

Description: The Delete Recipe endpoint allows users to delete the recipes they have created. This feature is restricted to authorised users who created the recipe.

Endpoint: http://127.0.0.1:8000/recipes/steps/create/

Methods: POST

Fields/Payload: step_num, cooking_time, prep_time, instructions

Example Response:


```
{
   "id": 3,
   "recipe_id": null,
   "step_num": 1,
   "cooking_time": "00:02:00",
   "prep_time": "00:02:00",
   "instructions": "instruct 2",
   "media": []
}
```


Additional Notes:

- **prep_time/cooking_time** must be in “00:00:00” format (i.e. HH:MM:SS)
- **step_num** is also updated when a new recipe is created based on the total number of steps in a recipe
- recipe_id is initially set to null and then will be updated when added to the Create a Recipe endpoint

3) Add Step(s) Media

Description: The Delete Recipe endpoint allows users to delete the recipes they have created. This feature is restricted to authorised users who created the recipe.

Endpoint: http://127.0.0.1:8000/recipes/steps/create/media/

Methods: POST

Fields/Payload: step_id, media

Response Example:


```
{
   "id": 1,
   "step_id": 1,
   "media": "http://127.0.0.1:8000/media/step04334.png"
}
```


Additional Notes:

- When none of the fields are filled out this response is sent:


```
{  "step_id": [
      "This field is required."
   ],
   "media": [
      "No file was submitted."
   ]}
```

4) Add Ingredients

Description: The Add Ingredients endpoint allows users (who have admin access) to create “parent” ingredients for other users to select when making a recipe.
Endpoint: http://127.0.0.1:8000/recipes/ingredients/create/
Methods: POST
Fields/Payload: name
Response Example:
{"id": 12, "recipe_id": null, "name": "Beef", "quantity": 0,   "unit": "cups" }
Additional Notes:

- name is required, if not provided you will get the following response:
    - {"name": ["This field is required.”]}
- duplicate ingredient names are not allowed, you will get the response:
    - {"name": ["Ingredient with this name already exists.”]}
- Ingredients with recipe_id = null represent the “parent” Ingredients where other Ingredients (more specific to a quantity and recipe_id) are made from in the Create Recipe endpoint. Ingredients where recipe_id = null can be searched through (or autofilled) to avoid duplicates.
- The default value for unit is “cups” to keep consistency in the code (as informed by a TA)

5) Create A Recipe

Description: The Create A Recipe view creates a recipe for the user. The fields user the previous steps to be done in order to fill out the required fields.
Endpoint: http://127.0.0.1:8000/recipes/create-recipe/

Methods: POST

Fields/Payload: name, difficulty, meal, diet, cuisine, cooking_time, prep_time, servings_num, media, steps, ingredients (*see additional notes for details)
Example Response:


```
{
  "id": 5,
  "user_id": 1,
  "name": "Banana Bread",
  "based_on": null,
  "total_reviews": 0,
  "total_likes": 0,
  "total_favs": 0,
  "published_time": "2023-03-13T21:02:59.962044Z",
  "difficulty": 2,
  "meal": 3,
  "diet": 3,
  "cuisine": 2,
  "total_time": "00:25:00",
  "cooking_time": "00:20:00",
  "prep_time": "00:05:00",
  "calculated_total_time": "01:20:00",
  "calculated_prep_time": "00:40:00",
  "calculated_cook_time": "00:40:00",
```

```
  "servings_num": 2,
  "media": [
    {
      "id": 3,
      "recipe_id": 5,
      "media": "/media/recipe-media/pic.png"
    }
  ],
  "steps": [
    {
      "id": 1,
      "recipe_id": 5,
      "step_num": 1,
      "cooking_time": "00:20:00",
      "prep_time": "00:20:00",
      "instructions": "instruction 1",
      "media": []
    }
  ],
  "ingredients": [
    {
      "id": 4,
      "recipe_id": 5,
      "name": "Bread",
      "quantity": 2,
      "unit": "cups"
    }
  ],
  "interactions": []
}
```


Additional Notes for Recipe Creation:

- **name** is a character field, **difficulty** is an int field from 0 to 2 where: (0, “‘Easy”), (1, “Medium”), (2 , “Hard”)
- **diet** is an int field from 0 to 5 where:(0, "Vegan"), (1, "Vegetarian"), (2, "Gluten-Free"), (3, "Halal"), (4, "Kosher"), (5, "None")
- **meal** is an int field from 0 to 5 where: (0, "Breakfast"), (1, "Lunch"), (2, "Dinner"), (3, "Desserts"), (4, "Snacks"), (5, "Other")
- **cuisine** is an int field from 0 to 13 where: (0, "African"), (1, "Caribbean"), (2, "East Asian"), (3, "European"), (4, "French"), (5, "Italian"), (6, "Middle-Eastern"), (7, "North American", (8, "Oceanic"), (9, "Russian"), (10, "Spanish"), (11, "South American"), (12, "South Asian"), (13, "Other")
- **prep_time/cooking_time** must be in “00:00:00” format (i.e. HH:MM:SS)
- **media** must be written in the format: 1, 2 where media id = 1 and media id = 2 (this is created in 1) Add Recipe Media)
- **steps** must be written in the format: 1, 2 where step id = 1 and step id = 2
- **ingredients** must be written in the format: {“1”: [2, “cups”], “2”: [3, “cups”} where the key values are the ingredient “id” and the first item in the list is the quantity and the second item is the unit
- Models linked to the RecipeModel will be deleted such as StepModel, InteractionModels, etc
- If any of the required fields (listed in Fields/Payload) are missing, the response with the missing field will be sent:
    - {"error": "servings_num is required."}

Recipe Detail

Description: The Recipe Detail view shows all users the view of the recipe.
Endpoint: http://127.0.0.1:8000/recipes/<recipe_id>/details/
Methods: GET
Fields/Payload: None
Example Response:


```
{
  "id": 5,
  "user_id": 1,
  "name": "Banana Bread",
  "based_on": null,
  "total_reviews": 0,
  "total_likes": 0,
  "total_favs": 0,
  "published_time": "2023-03-13T21:02:59.962044Z",
  "difficulty": 2,
  "meal": 3,
  "diet": 3,
  "cuisine": 2,
  "total_time": "00:25:00",
  "cooking_time": "00:20:00",
  "prep_time": "00:05:00",
  "calculated_total_time": "01:20:00",
  "calculated_prep_time": "00:40:00",
  "calculated_cook_time": "00:40:00",
  "servings_num": 2,
  "media": [
    {
      "id": 3,
      "recipe_id": 5,
      "media": "/media/recipe-media/pic.png"
    }
  ],
  "steps": [
    {
      "id": 1,
      "recipe_id": 5,
      "step_num": 1,
      "cooking_time": "00:20:00",
      "prep_time": "00:20:00",
      "instructions": "instruction 1",
      "media": []
    }
  ],
  "ingredients": [
    {
      "id": 4,
      "recipe_id": 5,
      "name": "Bread",
      "quantity": 2,
      "unit": "cups"
    }
  ],
  "interactions": []}
```

Edit Recipe

Description: The Edit Recipe view will only allow a user who created a recipe to edit their own recipe.
Endpoint: http://127.0.0.1:8000/recipes/<recipe_id>/
Methods: PATCH
Fields/Payload: name, difficulty, meal, diet, cuisine, cooking_time, prep_time, servings_num, media, steps, ingredients
Example Response:


```
{
    "id": 5,
    "user_id": 1,
    "name": "Banana Bread",
    "based_on": null,
    "total_reviews": 0,
    "total_likes": 0,
    "total_favs": 0,
    "published_time": "2023-03-13T21:02:59.962044Z",
    "difficulty": 2,
    "meal": 3,
    "diet": 3,
    "cuisine": 2,
    "total_time": "00:25:00",
    "cooking_time": "00:20:00",
    "prep_time": "00:05:00",
    "calculated_total_time": "01:20:00",
    "calculated_prep_time": "00:40:00",
    "calculated_cook_time": "00:40:00",
    "servings_num": 2,
    "media": [
        {
            "id": 3,
            "recipe_id": 5,
            "media": "/media/recipe-media/pic.png"
        }
    ],
    "steps": [
        {
            "id": 1,
            "recipe_id": 5,
            "step_num": 1,
            "cooking_time": "00:20:00",
            "prep_time": "00:20:00",
            "instructions": "instruction 1",
            "media": []
        }
    ],
    "ingredients": [
        {
            "id": 4,
            "recipe_id": 5,
            "name": "Bread",
            "quantity": 2,
            "unit": "cups"
        }
    ],
    "interactions": []
}
```

Additional notes:

- The fields listed in the Fields/Payload are optional

Delete a Recipe

Description: The Delete a Recipe view will only allow a user who created a recipe to delete their own recipe.

Endpoint: http://127.0.0.1:8000/recipes/<recipe_id>/delete/
Methods: DELETE
Fields/Payload: None
Example Response: Upon successful deletion: {"message": "Recipe has been deleted."}
Additional notes:

- If recipe_id is not valid then the response is sent:
  {"detail": "Not found."}
- If the recipe the user wants to delete is not theirs, the response is sent:
  {'message': 'You do not have permission to delete this recipe.'}

Remix a Recipe

Description: The Remix a Recipe view will only allow a user who remix an existing recipe.
Endpoint: http://127.0.0.1:8000/recipes/<recipe_id>/remix-recipe/
Methods: POST
Fields/Payload: name, difficulty, meal, diet, cuisine, cooking_time, prep_time, servings_num, media, steps, ingredients (*see additional notes for details)
Example Response:


```
{
  "id": 5,
  "user_id": 1,
  "name": "Banana Bread 2",
  "based_on":4,
  "total_reviews": 0,
  "total_likes": 0,
  "total_favs": 0,
  "published_time": "2023-03-13T21:02:59.962044Z",
  "difficulty": 2,
  "meal": 3,
  "diet": 3,
  "cuisine": 2,
  "total_time": "00:25:00",
  "cooking_time": "00:20:00",
  "prep_time": "00:05:00",
  "calculated_total_time": "01:20:00",
  "calculated_prep_time": "00:40:00",
  "calculated_cook_time": "00:40:00",
  "servings_num": 2,
  "media": [
    {
      "id": 3,
      "recipe_id": 5,
      "media": "/media/recipe-media/pic.png"
```

```
    }
  ],
  "steps": [
    {
      "id": 1,
      "recipe_id": 5,
      "step_num": 1,
      "cooking_time": "00:20:00",
      "prep_time": "00:20:00",
      "instructions": "instruction 1",
      "media": []
    }
  ],
  "ingredients": [
    {
      "id": 4,
      "recipe_id": 5,
      "name": "Bread",
      "quantity": 2,
      "unit": "cups"
    }
  ],
  "interactions": []
}
```


Additional Notes:

- All the fields listed in the Fields/payload are optional, when a remixed recipe is created the interactions will be an empty list, total_views, total_likes, total_favs, avg_rating will be set to 0

Published Recipes

Description: The Published Recipes view will show users all the recipes they have created.
Endpoint: http://127.0.0.1:8000/myrecipes/published-recipes/
Methods: GET
Fields/Payload: None
Example Response: A user with user_id 6 might have this response


```
{  "count": 1,
  "next": null,
  "previous": null,
  "results": [{
      "id": 6,
      "name": "Banana Bread 5.4",
      "difficulty": 2,
      "meal": 3,
      "diet": 3,
      "cuisine": 2,
      "cooking_time": "00:20:00",
      "avg_rating": 0.0,
      "total_reviews": 0,
      "total_likes": 0,
      "total_favs": 0,
      "media": "/media/recipe-media/ww12.png"}]}
```


Additional Details:

- Only the first image is displayed

Favourited Recipes

Description: The Favourite Recipes view will show users all the recipes they have marked as favourite.

Endpoint: http://127.0.0.1:8000/myrecipes/favourite-recipes/

Methods: GET

Fields/Payload: None

Example Response: A user with user_id 2 might have this response


```
{   "count": 1,
    "next": null,
    "previous": null,
    "results": [{
            "id": 2,
            "name": "Banana Bread 5.4",
            "difficulty": 2,
            "meal": 3,
            "diet": 3,
            "cuisine": 2,
            "cooking_time": "00:20:00",
            "avg_rating": 0.0,
            "total_reviews": 0,
            "total_likes": 0,
            "total_favs": 0,
            "media": "/media/recipe-media/ww12.png"}]}
```


Recent Recipes

Description: The Recent Recipes view will show users all the recipes they have interacted with (i.e. created, marked as favourite, rated, liked and commented).

Endpoint: http://127.0.0.1:8000/myrecipes/recent-recipes/

Methods: GET

Fields/Payload: None

Example Response: A user with user_id 2 will have this response


```
{  "count": 1,
   "next": null,
   "previous": null,
   "results": [{
            "id": 2,
            "name": "Banana Bread 5.4",
            "difficulty": 2,
            "meal": 3,
            "diet": 3,
            "cuisine": 2,
            "cooking_time": "00:20:00",
            "avg_rating": 0.0,
            "total_reviews": 0,
            "total_likes": 0,
            "total_favs": 0,
            "media": "/media/recipe-media/ww12.png"}]}
```

Home Page

Description: The Home Page view will show a preview of all popular recipes, as well as popular recipes in the, ‘breakfast’, ‘lunch’, and ‘dinner’ meal categories.
Endpoint: http://127.0.0.1:8000/recipes/
Methods: GET
Fields/Payload: None
Additional Notes:

- Returns a response which holds 4 querysets
    - ‘Popular’, a list of the first six recipes sorted by ‘total_reviews’
    - ‘Breakfasts’, a list of the first six breakfast recipes sorted by ‘total_favs’
    - ‘Lunches’, a list of the first six lunch recipes sorted by ‘total_favs’
    - ‘Dinners’, a list of the first six dinner recipes sorted by ‘total_favs’

Example Response: A GET request might give the following response:


```
{
  "Popular": [
    {
      "id": 9,
      "name": "eggs (multiple!!!!)",
      "difficulty": 1,
      "meal": 1,
      "diet": 5,
      "cuisine": 13,
      "cooking_time": "00:15:00",
      "avg_rating": 4.5,
      "total_reviews": 2,
      "total_likes": 2,
      "total_favs": 2,
      "media": null
    },
    {
      "id": 1,
      "name": "egg",
      "difficulty": 0,
      "meal": 0,
      "diet": 4,
      "cuisine": 0,
      "cooking_time": "00:00:01",
      "avg_rating": 0.0,
      "total_reviews": 0,
      "total_likes": 0,
      "total_favs": 0,
      "media": null
    },
    {
      "id": 10,
      "name": "another egg recipe",
      "difficulty": 2,
      "meal": 3,
      "diet": 5,
```

```
            "cuisine": 13,
            "cooking_time": "01:00:00",
            "avg_rating": 0.0,
            "total_reviews": 0,
            "total_likes": 0,
            "total_favs": 0,
            "media": "/media/recipe-media/egg_wveGSrV.jpg"
        }
    ],
    "Breakfasts": [
        {
            "id": 1,
            "name": "egg",
            "difficulty": 0,
            "meal": 0,
            "diet": 5,
            "cuisine": 13,
            "cooking_time": "00:00:01",
            "avg_rating": 0.0,
            "total_reviews": 0,
            "total_likes": 0,
            "total_favs": 0,
            "media": null
        }
    ],
    "Lunches": [
        {
            "id": 9,
            "name": "eggs (multiple!!!!)",
            "difficulty": 1,
            "meal": 1,
            "diet": 5,
            "cuisine": 13,
            "cooking_time": "00:15:00",
            "avg_rating": 4.5,
            "total_reviews": 2,
            "total_likes": 2,
            "total_favs": 2,
            "media": null
        }
    ],
    "Dinners": []
}
```

Recipe Search

Description: The Recipe Search view allows users to access a list of specific recipes based on their chosen filters.
Endpoint: http://127.0.0.1:8000/recipes/search/
Methods: GET
Fields/Payload: category, query, cuisine, meal, diet, cooking_time
Additional Notes:

- Instead of POST, this view uses a GET request with the filters being passed in through param fields
- The category field is required, and must be one of: ('Recipe', 'Ingredients', 'User')
- If the query field is not provided, the search will not filter based on the category.
- The cooking_time field expects an integer, and filters based on the integer:
    - 0: No filter
    - 1: filters for recipes under 10 minutes
    - 2: filters for recipes between 10-30 minutes
    - 3: filters for recipes between 30-60 minutes
    - 4: filters for recipes over 60 minutes
- If cuisine, meal, diet, or cooking_time are blank, then they will default to (cuisine=14, meal=6, diet=6, cooking_time=0)
- If the filter fields are the following values: (cuisine=14, meal=6, diet=6, cooking_time=0), then that filter will not be applied

Example Response: A GET request sent with the params (category: 'Ingredients', query: 'egg', diet: '5') might have the following response:


```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 9,
      "name": "eggs (multiple!!!!)",
      "difficulty": 1,
      "meal": 1,
      "diet": 5,
      "cuisine": 13,
      "cooking_time": "00:15:00",
      "avg_rating": 4.5,
      "total_reviews": 2,
      "total_likes": 2,
      "total_favs": 2,
      "media": null
    },
    {
      "id": 10,
      "name": "another egg recipe",
      "difficulty": 2,
      "meal": 3,
```

```
        "diet": 5,
        "cuisine": 13,
        "cooking_time": "01:00:00",
        "avg_rating": 0.0,
        "total_reviews": 0,
        "total_likes": 0,
        "total_favs": 0,
        "media": "/media/recipe-media/egg_wveGSrV.jpg"
    }
  ]
}
```


Autocomplete

Description: The Autocomplete view can help find a list of ingredients, users, or recipes based on a given query.
Endpoint: http://127.0.0.1:8000/recipes/autocomplete/
Methods: GET
Fields/Payload: category, query
Additional Notes:

- Instead of POST, this view uses a GET request with the filters being passed in through param fields
- The category field takes in an integer and searches through a specific model based on the value:
    - 0: Searches through IngredientModel based on name
    - 1: Searches through RecipeModel based on name
    - 2: Searches through User based on username
- If the category field is not provided, it defaults to 0 and searches based on ingredients
- If the query field is not provided, it defaults to a blank string and will return every object in the category

Example Response: A GET request sent with the params (category: 2, query: ‘dummy1’) might have the following response:


```
{
  "count": 3,
  "next": null,
  "previous": null,
  "results": [
    {
      "username": "testdummy100",
      "first_name": "test",
      "last_name": "dummy100"
    },
    {
      "username": "testdummy1001",
      "first_name": "test",
      "last_name": "dummy100"
    },
    {
```

```
        "username": "testdummy10",
        "first_name": "test",
        "last_name": "dummy10"
      }
  ]
}
```


Interaction Creation and Editing

Description: The Interaction view can create a new interaction, or edit an existing one.
Endpoint: http://127.0.0.1:8000/recipes/<recipe_id>/details/interaction/
Methods: POST, PATCH
Fields/Payload: rating, comment, like, favourite
Additional Notes:

- Every interaction affected by this view will be between the currently authenticated user and the recipe with id <recipe_id> in the url
- To create a new interaction, send a POST request. If an interaction already exists, the response will tell the user to send a PATCH request instead.
- To update an existing interaction, send a PATCH request. If no interaction exists, the response will tell the user to send a POST request instead.
- If any of the fields are not provided, they will default to:
    - rating: 0.0
    - comment: Blank
    - like: False
    - favourite: False
- The rating and comment fields must be provided together; they must both be blank, or neither can be blank.

Interaction Media Creation

Description: The Add Interaction Media view can create a new ReviewMedia object, and link it to an existing interaction.
Endpoint: http://127.0.0.1:8000/recipes/interactions/<interaction_id>/add-media/
Methods: POST
Fields/Payload: media
Additional Notes:

- The media field is required, and must be an image file.
- The ReviewMedia object created will be linked to the InteractionModel object with <interaction_id> id
    - If no such interaction exists, an error response will be returned with code 404.

All Recipes

Description: Displays a list of all recipes that have been created along with information about each recipe. The information includes each recipe's name, difficulty, meal, cuisine, total_reviews, total_likes, total_favs.
Endpoint: http://127.0.0.1:8000/recipes/all-recipes/
Methods: GET
Fields/Payload: None
Example Response:


```
{  "count": 2,
   "next": null,
   "previous": null,
   "results": [
      {       "name": "Recipe A",
         "difficulty": 0,
         "meal": 5,
         "cuisine": 13,
         "total_reviews": 0,
         "total_likes": 0,
         "total_favs": 0
  },

      {       "name": "Recipe B",
         "difficulty": 0,
         "meal": 5,
         "cuisine": 13,
         "total_reviews": 0,
         "total_likes": 0,
         "total_favs": 0
      }
]}
```


Additional Notes:

- This endpoint does *not* require the user to be authenticated, that is, any user logged in or not can access this endpoint.
- This endpoint is paginated by a default value of 10 items per page. References to previous and the next pages are indicated by the fields "previous" and "next" in the response.

Popular Recipes

Description: Displays a list of the most popular recipes based on a filter provided by the url parameter filter. Users can filter recipes based on the total number of reviews, likes, and favourites.
Endpoint: http://127.0.0.1:8000/recipes/popular/<str:filter>/
Methods: GET
Fields/Payload: filter
Example Response: A GET request sent with the params (filter='total_likes') might have the following response:

```
{
  "count": 3,
  "next": null,
  "previous": null,
  "results": [
    {  "name": "Recipe C",
       "difficulty": 0,
       "meal": 5,
       "cuisine": 13,
       "total_reviews": 2,
       "total_likes": 21,
       "total_favs": 13
    },
    {   "name": "Recipe B",
       "difficulty": 1,
       "meal": 5,
       "cuisine": 13,
       "total_reviews": 7,
       "total_likes": 13,
       "total_favs": 5
    },
    {   "name": "Recipe A",
       "difficulty": 2,
       "meal": 5,
       "cuisine": 13,
       "total_reviews": 2,
       "total_likes": 4,
       "total_favs": 9
    }
  ]}
```


Additional Notes:

- This endpoint does *not* require the user to be authenticated
- Results are displayed in descending order.
- filter must be one of: [“total_reviews”, “total_likes”, “total_favs”]
    - If any other value for filter is provided, the following error response is returned:

      { "detail": "Not a valid filter. Please select a filter from the following list: ['total_reviews', 'total_likes', 'total_favs']"}

- This endpoint is paginated by a default value of 10 items per page. References to previous and the next pages are indicated by the fields “previous” and “next” in the response.

User Shopping Cart:

Whenever a user creates a recipe, or interacts with any recipe by (liking, favouriting, or commenting), the recipe is added to the user's shopping cart by creating an instance in the ShoppingRecipeModel.

Shopping Cart By Individual List of Recipes

Description: Displays a list containing information about each recipe the logged in user has interacted with. The information includes the recipe_id, name, servings_num (which is the serving size of the recipe the user has selected), and a list of ingredients which contains each ingredient's name and quantity.
Endpoint: http://127.0.0.1:8000/accounts/shopping-list/
Methods: GET
Fields/Payload: None
Example Response:


```
[
  {         "recipe_id": 1,
    "name": "Recipe A",
    "servings_num": 2,
    "ingredients": [
      {"name": "Ingredient A", "quantity": 4},
      {"name": "Ingredient A2","quantity": 6}]
  },
  {         "recipe_id": 2,
    "name": "Recipe B",
    "servings_num": 2,
    "ingredients":
                  [{"name": "Ingredient B", "quantity": 7},
                  {"name": "Ingredient A","quantity": 1}]
  }
]
```


Additional Notes:

- This endpoint requires the user to be authenticated, that is, logged in.
- The quantity of each ingredient for every recipe is determined based on the serving size chosen.
- If no recipes exist, or if the user has interacted with no recipes, the following response is returned: []

Shopping Cart By Combined List of Ingredients

Description: Displays a list containing the name and quantity of every ingredient for every recipe the user has interacted with.

Endpoint: http://127.0.0.1:8000/accounts/combined-list/
Methods: GET
Fields/Payload: None
Example Response:

```
[
  {"name": "Ingredient A", "quantity": 5},
  {"name": "Ingredient A2", "quantity": 6},
  {"name": "Ingredient B", "quantity": 7}
]
```

Additional Notes:

- This endpoint requires the user to be authenticated, that is, logged in.
- The quantity of each ingredient is determined based on the serving size chosen.
- If multiple recipes share the same ingredients, the shared ingredients are only listed once but contain a sum of all the quantities.
- If no recipes exist, or if the user has interacted with no recipes, the following response is returned: []

Updating The Serving Size Of A Recipe

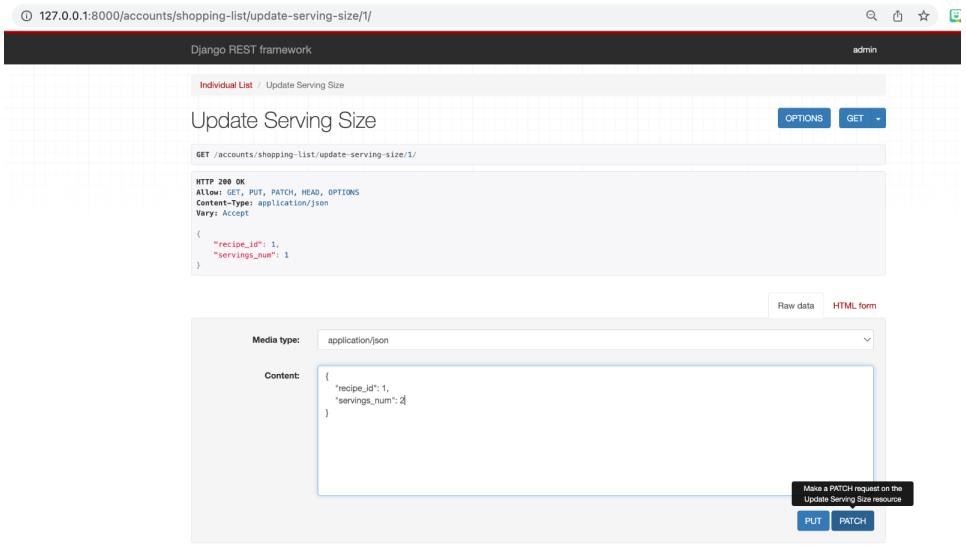
Description: Allows users to view and edit the serving size of a given recipe, which modifies the quantities of the recipe's ingredients in the shopping cart.

Endpoint: http://127.0.0.1:8000/accounts/shopping-list/update-serving-size/<int:recipe_id>/
Methods: GET, PATCH
Fields/Payload: recipe_id
Additional Notes:

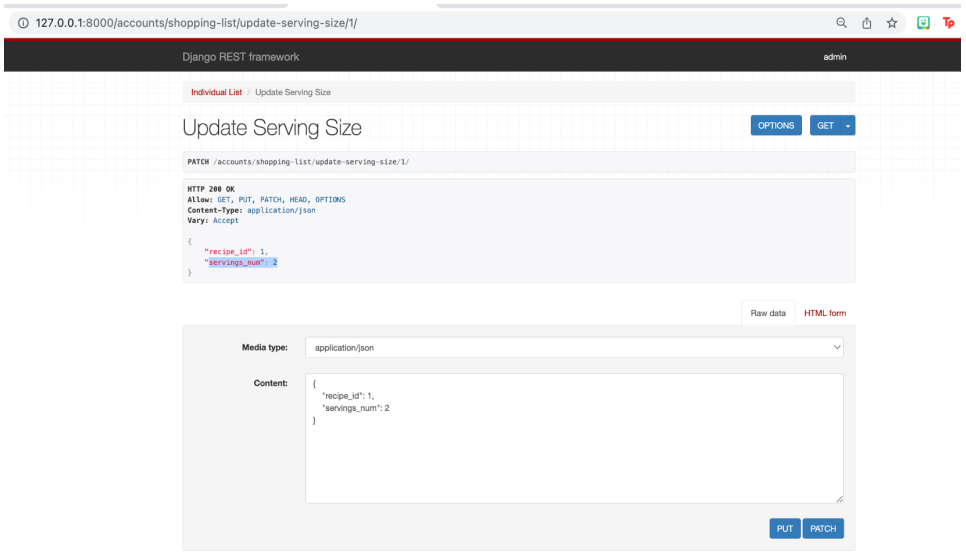
- This endpoint requires the user to be authenticated, that is, logged in.
- On a GET request, the recipe_id and servings_num will be returned.
- To modify the servings_num, send a PATCH request with the recipe_id of the recipe whose serving size you want to modify in your shopping cart, and the updated value for servings_num. Below an example is shown where we update servings_num = 2 for recipe_id = 1.
- If no recipe_id from the url parameter exists, a 404 Not Found error is returned.

GET Request and Response Example:

- Note: The servings_num value was changed from 1 to 2, then we click the PATCH button to submit the PATCH request



Patch Request Response Example:



Remove Recipe From Shopping Cart

Description: Removes a recipe from the logged in User's shopping cart.
Endpoint: http://127.0.0.1:8000/accounts/shopping-list/remove/<int:recipe_id>/
Methods: DELETE
Fields/Payload: recipe_id

Additional Notes:

- This endpoint requires the user to be authenticated, that is, logged in.
- If no recipe_id from the url parameter exists, a 404 Not Found error is returned with the following response: {"detail": "Not found."}
- On success, the following response is returned:
  { "message": "Recipe has been deleted."}

Empty Shopping Cart

Description: Empty the logged in User's shopping cart.
Endpoint: http://127.0.0.1:8000/accounts/shopping-list/clear/
Methods: DELETE
Fields/Payload: None
Additional Notes:

- This endpoint requires the user to be authenticated, that is, logged in.
- On success, the following response is returned:
  {"message": "Shopping Cart Cleared"}
- Note: the same response is returned whether or not the shopping list contained any recipes or not.