ENDPOINT DOCUMENTATION

RECIPE ENDPOINTS:

GET /recipes/{recipe_id}
This endpoint returns a single recipe by its identifier. For each recipe
it returns:
* `recipe_id`: the internal id of the recipe. Can be used to query the
`/recipes/{id}` endpoint.
* `recipe_name`: The name of the recipe.
* `cuisine`: The cuisine that the recipe is from.
* `meal_type`: The meal type that the recipe is from (breakfast, lunch, dinner, etc).
* `ingredients`: The listed ingredients and amounts that are needed to make the recipe.
* `prep_time_mins`: The total time needed to prep the recipe in minutes.
* `instructions`: The instructions needed to make the recipe.
* `number_of_favorites`: The number of users that have favorited the recipe.

GET /recipes/
This endpoint returns a list of recipes. For each recipe it returns:
* `recipe_id`: the internal id of the character. Can be used to query the
`/recipes/{recipe_id}` endpoint.
* `recipe`: The name of the recipe.
* `cuisine`: The cuisine that the recipe is from.
* `meal_type`: The meal type that the recipe is from.
* `time`: time needed to make the recipe.
* `instructions`: instruction to make the recipe.
* `number_of_favorites`: number of users who have favorited recipe.

You can filter for recipes whose name contains a string by using the
`recipe` query parameter. You can filter for recipes by cuisine by using the
`cuisine` query parameter. You can filter for recipes by meal-type by using the
`type` query parameter.

You can also sort the results by using the `sort` query parameter:
* `recipe` - Sort by recipe name alphabetically.
* `time` - Sort by cooking time.
* `number_of_favorites` - Sort by number of users who have favorited recipe.

The `limit` and `offset` query parameters are used for pagination.
The `limit` query parameter specifies the maximum number of results to return.
The `offset` query parameter specifies the number of results to skip before
returning results.

POST /recipes/
This endpoint will allow users to add their own recipes to the API.
To add a recipe, the user must provide:
* `recipe`: The name of the recipe.
* `cuisine_type_id`: The cuisine id(s) for the recipe cuisine(s).
* `meal_type_id`: The meal type id(s) for the recipe meal type(s).
* `calories`: Total calories in one serving of the recipe.
* `time`: The total time it takes to make the recipe.
* `recipe_instructions`: The instructions to make the recipe.
* `url`: The url to the recipe.
* `ingredients`: The list that contains the ingredients and amounts that are needed to make the recipe.

PUT /favorited_recipes/
This endpoint will allow users to add existing recipes to their favorites list.
To favorite a recipe, the user must provide:
* `user_id`: The name of the recipe.
* `recipe_id`: The cuisine id(s) for the recipe cuisine(s).

DELETE /favorited_recipes/
This endpoint will allow users to remove existing recipes from their favorites list.
To unfavorite a recipe, the user must provide:
* `user_id`: The name of the recipe.
* `recipe_id`: The cuisine id(s) for the recipe cuisine(s).

GET /favorited_recipes/
This endpoint will list all recipes in the users favorites list. For each recipe it returns:
* `recipe_id`: the internal id of the character. Can be used to query the `/recipes/{recipe_id}` endpoint.
* `recipe`: The name of the recipe.
* `cuisine`: The cuisine that the recipe is from.
* `meal_type`: The meal type that the recipe is from.
* `prep_time_mins`: The time needed to make the recipe.
* `instructions`: The instructions for making the recipe.
* `ingredients`: The listed ingredients and amounts that are needed to make the recipe.
* `prep_time_mins`: time needed to make the recipe.

The `limit` and `offset` query parameters are used for pagination.
The `limit` query parameter specifies the maximum number of results to return.
The `offset` query parameter specifies the number of results to skip before

INGREDIENT ENDPOINTS:

GET /ingredients/{ingr_id: int}
This endpoint returns a single ingredient by its identifier. For each ingredient it returns:
* `ingredient_id`: the internal id of the ingredient. Can be used to query the `/ingredients/{id}` endpoint.
* `ingredient`: The name of the ingredient
* `recipes`: A list of the recipes that contain the ingredient.

POST /ingredients/
This endpoint adds a single ingredient. A new ingredient is represented by its name. This endpoint will return the new id of the ingredient created.
To add a new ingredient, the user must provide:
* `ingredient_name`: The name of the ingredient to add.

USER ENDPOINTS:

POST /register_user/
This endpoint will allow users to create a new user. The user must provide a username and password. This endpoint check to make sure no duplicate usernames can be used. All passwords will be stored as hashes to protect user privacy. This endpoint will return the user_id of the newly created user.
To register a new user, the user must provide:
* `username`: The username of the user to add.
* `password`: The password of the user to add.

POST /login_user/
This endpoint will allow users to check if their password is valid. This endpoint will throw an error if the user does not exist, or if the password is incorrect.
To validate the login information of an existing user, the user must provide:
* `username`: The username of the user.
* `password`: The password of the user.