General Rules:

Codes:

* 1: Request Successful
* 0: Simple Error
  + Usually print error as popup
* -1: Bad Request, not good

GENERAL CALLS

/getRecipes

This call occurs first right when app starts. Gets first 30 recipes from the database for the home page

* HTTP Type: GET
* Requirements
  + None
* Response output
  + {recipes: [] }
    - a json object with a recipes array

/getRecipeByID

This call occurs on the start of each recipe page being loaded on the browser. It gathers all the recipe data based on the recipe id. To populate the page with.

* HTTP Type: GET
* Requirements (url param)
  + ID -> recipe integer id
* Response output
  + {name,prep\_time,cooking\_time,style\_id,image\_location,instruction,ingredients [] }
    - all the data that the recipe holds from the database, including an ingredients array of jsons {ingredient\_name,quantity}

/createUser

This call is when you want to create a new user. This call will handle if the username is taken. Maybe later that the user email is registered.

* HTTP Type: POST
* Requirements
  + user\_name
  + user\_password
  + user\_email -> email of user
  + firstname
  + lastname
* Response output
  + {code,message,data}
    - ex. code:1, message: user created, data:{}

/login

This call is when the user wants to login. The call will return a session token to use for other calls.

* HTTP Type: POST
* Requirements
  + user\_name
  + user\_password
* Response output
  + {code,message, data: {user\_id,user\_token}}
    - ex. {code: 1, message: Login Successful, data: {user\_id,user\_token}}
    - the server will generate a user token and pass that as a data json inside the response json

/getComments

This call is for getting all comments on a recipe

* HTTP Type: GET
* Requirements (url params)
  + recipe\_id <- int value
* Output
  + {comments: [{user,comment},...]
    - comments json array made of user,comment jsons

ACTION CALLS

/logout

This call is when the user wants to log off

* HTTP Type: POST
* Requirements
  + user\_token -> user specific key
* Response output
  + {code, message,data}
    - ex. code:1,message:logout successful, data:{}

/addComment

This call is for authenticated users to post comments on recipes

* HTTP Type: POST
* Requirements
  + user\_id
  + recipe\_id
  + message
  + user\_token
* Response output
  + {code,message,data}
    - data is null

/getUserData

Call for auth users to pull data from their account

* Http Type: POST
* Requirements
  + user\_token
  + user\_id
* Response
  + {code,message,data:{user\_id,username,email,first\_name,last\_name,biography}}
    - all data except password is provided

/addRecipe

User can add a recipe to recette website

* Http Type: POST
* Requirements
  + user\_id
  + user\_token
  + recipe\_data:{user\_id,name,prep\_time,cooking\_time,ready\_in,origin,instruction,image\_location}
* Response
  + {code,message,data:null}

/linkIngredients

When user adds ingredients to a Recipe, this will be called

* HTTP Type: POST
* Requirements
  + recipe\_id
  + ingredient\_id
  + quantity
  + unit
* Response
  + {code,message,data:null}

/getUnits

Gets all units available in database

* HTTP Type: GET
* Requirements
  + None
* Response
  + {Units: []} (json file with all the units)

/addActivity

Users can comment and like/favorite someones recipe

* HTTP Type: POST
* Requirements
  + user\_id
  + message
* Response
  + {code, message, data:null}

/addPreferences

User can pick and choose preferences on what types of Recipes he wants to see

* HTTP Type: POST
* Requirements
  + user\_id
  + style\_id
* Response
  + {code, message, data:null}

/addFavorite

User can add certain Recipes to their favorites

* HTTP Type: POST
* Requirements
  + user\_id
  + recipe\_id
* Response
  + {code, message, data:null}

/addStyle

When a certain style is not available, server should add it

* HTTP Type: POST
* Requirements
  + name
* Response
  + {code, message, data:null}

/addIngredient

When an ingredient is not in DB, server should add

* HTTP Type: POST
* Requirements
  + name