Universal Rules:

Codes:

* 1 : Request successful, everything went well
* 0 : Request successful, simple error
  + Usually, just print the error as a Toast Message
* -1 : BAD Request , restart the app back to the login screen and have the user re-login(preferably this occurs automatically)

GENERAL CALLS

All calls that don’t relate to the matching/college parking stuff.

/codes

This call should occur first right when the app starts, before anything. I will also have in check the app’s version , but for now you can just make the call.

* HTTP Type: Get
* Requirements
  + None
* Output
  + {general\_key : \_\_\_\_\_\_ }
    - Use the general key for future calls

/forgot

This call when the user forgets their username/password. User should input their email and what info to get. When username is forgotten, their username is emailed to them. When their password forgotten, it is reseted and emailed to them.

* HTTP Type: Post
* Requirements
  + token\_gen -> general key
  + user\_email -> email of user
  + type\_forget -> what did the username forget. Only possible values:
    - username
    - password

/createUser

This call when the user creates a new user. This call will handle is the username is taken or if the user email already is registered. No promo user for now

* HTTP Type: Post
* Requirements
  + token\_gen -> general key
  + user\_name
  + user\_password -> no need to encrypt it
  + user\_email -> email of user

/updateUser

This call when the user updates their info within the app. THE APP MUST ask and check the user’s current password if they are changing it. The call must include the thing they want to change

* HTTP Type: Post
* Requirements
  + token\_gen -> general key
  + user\_name
  + user\_password
  + update\_type -> what does the user want to update
    - username
    - password

/login

This call when the user wants to login in. The call will return the user’s id and token to use for future call

* HTTP Type: Post
* Requirements
  + token\_gen -> general key
  + user\_name
  + user\_password -> no need to encrypt it
* Output:
  + {code:1, message : “User logged In”, data ; {user\_token : \_\_\_\_\_\_\_\_, user\_id : \_\_\_\_\_\_ }}

/addSuggestion

This call when the user wants to register a suggestion

* HTTP Type: Post
* Requirements
  + token\_user -> user specific key
  + user\_id -> user’s user\_id
  + type -> type of suggestion
  + system\_data
  + comment

/Logoff

This call when the user wants to log off

* HTTP Type: Post
* Requirements
  + token\_user -> user specific key
  + user\_id -> user’s user\_id

Action Calls

All calls that relate to the matching/college parking stuff.

The way these calls work, you just have to make one single http call ; based on the action the user performs within the app, the call’s details need to change

Base:

/action

This call when the user performs an action

* HTTP Type: Post
* Requirements
  + token\_user -> user specific key
  + user\_id -> user’s user\_id
  + action

….

Here are the possible actions:

* getUserStatus
  + Use this to get the user’s status
  + The app will make this call, and based off of the result will call an action/fragment
* updateLocation
  + Use this action to update the user’s location
  + Additional Parameters:
    - Lat
    - Lng
* request
  + Use this action to place a request for a park / ride
  + Additional Parameters:
    - College\_id
    - Parkinglot\_id
    - Pu\_lat
    - Pu\_lng
    - Type :
      * Ride
      * Spot
* cancelRequest
  + Use this action to cancel a request
* cancelMatch
  + Use this action to cancel a current match
* rateMatch
  + use this action to rate a match
  + Additional Parameters:
    - rating
* confirm
  + Use this action for when the user clicks ‘confirm’ in a match