UPDATE PROFILE TUTORIAL

Composed by: Charlie Tan

Local Route: http://localhost:5000/api/update_profile (update_profile)

Deployed Route: https://kindling-lp.herokuapp.com/api/update_profile (update_profile)

EXPECTED INPUT FROM FRONTEND:

```
{
  "email_str" : some_string,
  "update_fields_obj" : { some_obj_of_varying_number_of_key_value_pairs },
  "access_token_str" : some_string
}
```

INPUT PROPERTIES EXPLAINED:

1) **email_str**: email string corresponding to the user whose profile we are going to update.
2) **update_fields_obj**: a JSON object with a variable number of key-value pairs. The key-value pairs dictate which profile field is to be updated, and to what.
3) **access_token_str**: the golden ticket that serves to prove that the client is the client. For a more in-depth explanation on access tokens, please refer to the login API tutorial.

UPDATE_FIELDS_OBJ EXPLAINED:

-There are a total of 3 fields that a user can possibly update in their profile. These are:

  1) display name
  2) phone number
  3) description

-Thus, the 'update_fields_obj' can take a maximum of 3 key-value pairs.

-For example, if you wanted to update all 3 fields, you would structure 'update_fields_obj' like this:

```
{
  "display_name_str" : "some_display_name",
  "phone_str" : "some_phone_number",
  "description_str" : "some_description"
}
```

The result of the above would be that the user's display name would be updated to 'some_display_name', the user's phone number would be updated to 'some_phone_number', and finally, the user's description would be updated to 'some_description'.

-Note, that the order you put the key-value pairs in does not matter.

For example, the following 'update_fields_obj' would achieve the same thing as the one illustrated above:

```
{
  "description_str" : "some_description",
  "phone_str" : "some_phone_number",
  "display_name_str" : "some_display_name"
}
```

-If you wanted to only update 1 profile field, you would structure 'update_fields_obj' like this:

```
{
  "phone_str" : "another_phone_number"
}
```

The result of the above would be that ONLY the user's phone number would be updated to 'another_phone_number'.

-If you wanted to update 2 fields only, you would only provide the 2 key-value pairs that you wished to update in 'update_fields_obj':

```
{
  "display_name_str" : "awesome_display_name",
  "description_str" : "awesome_description"
}
```

The result of the above would be that only the user's display name and description would be updated to 'awesome_display_name' and 'awesome_description' respectively.

EXPECTED OUTPUT FROM BACKEND:

```
{
  "success_bool" : some_boolean,
  "refreshed_token_str" : some_string
}
```

OUTPUT PROPERTIES EXPLAINED:
1) **success_bool**: whether or not the user's profile was successfully updated. If 'true', the user's profile was updated. 'false' otherwise.
2) **refreshed_token_str**: the refreshed access token that is provided by the backend upon a successful API call. For more information on access tokens, please reference the login API tutorial.

-I usually place an 'expected output illustrated' section here, but I am not going to for this API.

-Some of you seemed to think we needed to alert the user to each and every possible outcome permutation, but we really don't need to air our dirty laundry to the whole world do we? (do we really need to let the user know that our database had a connection error?).

-Just focus on whether or not the user's profile was updated. This is a binary outcome, yes or no, true or false. This will give you less to worry about, and simplify your API integration code.

-If after considering all of this, you still feel you require more information from the output of this API, let me know and we'll figure it out.