LOGIN API TUTORIAL

Composed by: Charlie Tan

Local Route: http://localhost:5000/api/login
Deployed Route: https://kindling-lp.herokuapp.com/api/login

EXPECTED INPUT FROM FRONTEND:

```
{
  "email_str" : some_string,
  "password_str" : some_string
}
```

INPUT PROPERTIES EXPLAINED:

1) email_str: the email string that the client enters as their username.
2) password_str: the password string that the client enters as their password.

EXPECTED OUTPUT FROM BACKEND:

```
{
  "success_bool" : some_string,
  "email_str" : some_string,
  "is_group_bool" : some_boolean,
  "access_token_str" : some_string
}
```

OUTPUT PROPERTIES EXPLAINED:

1) success_bool: whether or not the database had an entry with both 'email_str' and 'password_str' in it. If 'true', there was such an entry in the database and we can consider the client logged in. 'false' otherwise.
2) email_str: the email string that the client entered as their username.
3) is_group_bool: whether or not the logged in client is considered a 'group' or an 'individual'. If 'true', the client is considered a 'group'. 'false' otherwise.
4) access_token_str: if login is successful, this is the string that serves to prove that the client is the client.

EXPECTED OUTPUT ILLUSTRATED:

1)
-Case: client logs in successfully.
-Expected output:

```
{
  "success_bool" : true,
  "email_str" : "the_email_of_the_logged_in_client",
  "is_group_bool" : true or false,
  "access_token_str" : "some_encoded_string"
}
```

2)
-Case: client provides bad credentials.
-Expected output:

```
{
  "success_bool" : false,
  "email_str" : "the_email_used_in_the_attempted_login",
  "is_group_bool" : "",
  "access_token_str" : ""
}
```

-Notice in the above case for bad credentials, that 'is_group_bool' shows an empty string (""). This is because in the case of a failed login, neither 'true' or 'false' would apply here. However, if you really wanted to, empty string is considered a 'falsey' value in JavaScript. Therefore, you could hypothetically do something like if(some_string), or if(!some_string) and it would work.

-In the above case for bad credentials, 'access_token_str' is also showing an empty string due to the fact that we do not provide access tokens for failed log-ins.

NOTES ON 'ACCESS TOKEN STRING':
-This string serves as the golden ticket for future server requests, so it is VERY important! If a future API call is made without providing this access token, the request is denied. When you get the initial acccess token returned from a successful login API call, please save it for the future.
-To WebApp people: 'tokenStorage.js' would be useful here, so look through it! This is located in frontend/src.
-Most future API calls will require providing the access token, however, there are a few that do not (can be seen in Peyton's API Google Doc).
-When a future, post-login API call is made successfully, the backend will provide a 'refreshed' access token. This 'refreshed' token is necessary because the access tokens actually have an expiration date (defaulted to 20 minutes).
-If for some reason the access token gets tampered with (you are not who you say you are), or the token expires, the server request would not work and the client would have to login again.
