GET READY STATUS TUTORIAL
Composed by: Charlie Tan

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

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

EXPECTED INPUT FROM FRONTEND:

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

INPUT PROPERTIES EXPLAINED:
1) email_str: the email string corresponding to the user whose 'ready_status' code is to be found.
2) access_token_str: this is the access token originally provided by the backend proving that you are who you say you are. For more information on access tokens, please reference the login API tutorial.

EXPECTED OUTPUT FROM BACKEND:

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

OUTPUT PROPERTIES EXPLAINED:
1) success_bool: whether or not the 'ready_status' could be found. If 'true', the 'ready_status' was able to be found. 'false' otherwise.
2) ready_status_int: the 'ready_status' code corresponding to the user whose email is given in 'email_str'.
3) 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.

EXPECTED OUTPUT ILLUSTRATED:

1)
-Case: the 'access_token_str' provided by the frontend input is invalid (expired or tampered

with).
-Expected output:
{
  "success_bool" : false,
  "ready_status_int" : -1234,
  "refreshed_token_str" : ""
}

-In this case, the database is never checked due to receiving a bad token. Notice how the backend returns empty string ("") upon receiving an invalid access token. Server requests will fail from here on out using this bad token. Only thing to do here is redirect back to the login page so the client can login again for a new, valid access token.

2)
-Case: the provided access token is valid, but a user with an email corresponding to 'email_str' from the frontend input could not be found in the database somehow.
-Expected output:
{
  "success_bool" : false,
  "ready_status_int" : -1234,
  "refreshed_token_str" : some_string_representing_a_refreshed_token
}

-In this case, since we check the database, we know that we have already passed the security procedure involving the access token. Therefore, we signal that the search for the 'ready_status' failed with a 'false' boolean, BUT we still send back a valid, refreshed access token. Be sure to save this new token!

3)
-Case: the provided access token is valid and we were able to find the 'ready_status' code for the user with an email corresponding to 'email_str' from the frontend input.
-Expected output:
{
  "success_bool" : true,
  "ready_status_int" : some_integer,
  "refreshed_token_str" : some_string_representing_a_refreshed_token
}

-Remember to save this new token!
