GET Requests

GET user by id `/api/users/<int:id>/`
URL: id (int)
PARAMETERS: none/empty
success_response is dictionary with user_id (int), name (String), username (String), email (String), favorites (dictionary of ids),

GET location by id `/api/locations/<int:id>/`
URL: id (int)
PARAMETERS: none/empty
success_response is location.serialize()

GET comments_by_location `/api/comments/<int:location_id>/`
URL: location_id (int)
PARAMETERS: none/empty
success_response is dictionary with id (of comment itself) (int), text (String), number (int), user_id (int), location_id (int), time_stamp (String), expiration (Boolean)

GET get_image `/api/users/images/<int:user_id>/`
URL: user_id (int)
PARAMETERS: none
success_response contains dictionary with url (string)

GET get user position by user id `/api/positions/<int:user_id>/`
URL: user_id (int)
success_response contains list of position.serialize()
→ Not currently used by Frontend, but useful for planned features of the app
i.e. using history of User positions to predict future busyness

POST Requests

AUTHENTICATION

POST register_user `/api/users/`
BODY: name (String), username (String), email (String), password (String)
success_response contains user_id, session_token, session_expiration, update_token

POST login `/api/users/login/`
BODY: email (String), password (String)
success_response contains user_id (integer), session_token, session_expiration, update_token (Strings for last two)

POST logout `/api/users/logout/`
HEADER Authorization: Bearer {session_token}
success_response contains a message indicating success of logout.

IMAGES
POST upload `/api/users/upload/<int: user_id>/`
URL: user_id (int)
BODY: image_data (image_data_type)
success_response contains asset.serialize()

OTHER (THE REST)
POST update_session `/api/session/`
HEADER: Bearer {session_token}
PARAMETERS: none/empty
BODY: none/empty
success_response contains {session_token, session_expiration, update_token}

POST add_comment `/api/comments/<int:location_id>/`
URL: location_id (int)
BODY: user_id (int), text (String) [optional], number (int) [required], latitude (float), longitude (float)
success_response contains dictionary with id (int), text (String), number (int), timestamp (String)
→ Creates one-to-many relationship between User/Location and Comment

POST update_busyness `/api/locations/busyness/<int:location_id>/`
URL: location_id (int)
success_response contains dictionary with busyness (float)

POST add_favorite `/api/favorites/<int:location_id>/`
URL: location_id (int)
BODY: user_id (int)
success_response contains location.simple_serialize()
→ Creates many-to-many relationship between User and Location (favorites for the user)

POST remove_favorite `/api/favorites/<int:location_id>/remove/`
URL: location_id (int)
BODY: user_id (int)
success_response contains user.serialize()

POST add_position `/api/positions/<int:user_id>/`
URL: user_id (int)
PARAMETERS: latitude (float), longitude (float)

success_response contains dictionary with id (int), user_id (int), latitude (float), longitude (float) timestamp (String)
→ Not currently used by Frontend, but useful for planned features of the app
i.e. using history of User positions to predict future busyness

DELETE Requests

DELETE delete_comment `/api/comments/`
HEADER: Bearer {session_token}
BODY: comment_id (int)
success_response contains comment.simple_serialize()

DELETE delete_user `/api/users/`
HEADER: Bearer {session_token}
BODY: none/empty
success_response contains user.simple_serialize()