Backend Documentation

<u>How to start and run our back-end:</u>

```
`python3 -m venv venv
source venv/bin/activate
pip3 install -r requirements.txt
flask init-db
flask --debug run`
```

<u>Sessions</u>

Sessions will be added in future sprints as a way for logged in users to access their profile and use the system. This has now been implemented

<u>Methods</u>

get_db()

Params: None

Return: g.db (the database)

close_db()

Params: e=None

Return: Nothing (closes database)

init_db()

Params: None

Return: Nothing (initializes the database)

init_db_command()

Params: None

Return: Nothing (clears existing data and sets up database)

init_app(app)

Params: app

Return: Nothing (registers database handling functions with the app)

__init__()

Params: self, groupID, name, activity

Return: Nothing (initializes the group)

search()

Params: self, groups, activity

Return: group (the groups that match the searched activity)

route()

Params: None

Return: returns “Hi” (just a test route)

example()

Params: None

Return: {‘messages’: list(map(dict, messages))}

login()

Params: None

Return: returns redirect(‘/’)

logout()

Params: None

Return: returns redirect(‘/’)

register()

Params: None

Return: returns redirect(‘/register’)

Create_Group()

Params: None

Return: None

__init__()

Params: self, firstName, lastName, email, password, address

Return: Nothing (initializes a user)

getFirstName()

Params: self

Return: self.firstname

getLastName()

Params: self

Return: self.lastname

getEmail()

Params: self

Return: self.email

getPassword()

Params: self

Return: self.password

getAddress()

Params: self

Return: self.address

loginUser()

Params: email, password, db

Return: User(firstName, lastName, email, password, address

logoutUser()

Params: email, db

Return: None

registerUser()

Params: firstName, lastName, email, password, address, db

Return: new_usr

account_information()

Params:

Returns: {‘messages’: info}

join_group()

Params:

Returns: None

edit_account()

Params:

Returns: temp

account_info_authentication()
Params: None
Returns: temp

group_info()
Params: group_id
Returns: jsonify(group)

contact_us()
Params:
Returns: {'messages':1}

API Endpoint Documentation:

- /user
    - GET: returns the current session
- /user_in_group/<group_id>
    - GET: returns information about the current user’s membership status in a group
- /login
    - POST: takes in an email and password and logs the user in with it
- /getGroupInfo
    - GET: returns information about a group with the given group ID
- /get_Friend_Message
    - GET: returns data about a friend’s messages
- /logout

    - POST: logs out the current user
- /updateAccount
    - POST: updates account information for a given user
- /deleteAccount
    - DELETE: deletes the given user
- /register
    - POST: registers a new user using the given information
- /search
    - POST: returns a list of groups matching the given criteria
    - GET: returns a list of all groups
- /Create_Group
    - POST: creates a group with the given information
    - GET: returns a list of activities
- /group_pic/<group_id>
    - GET: returns the group picture of the group with the given ID
- /set_group_pic
    - POST: sets the group picture of a group to the given file
- /get_acts
    - GET: returns a list of available activities
- /tracking/<user_id>
    - GET: returns a list of activities the given user is following
- /get_matching_users/<activity_id>/<user_id>
    - GET: returns a list of users (other than the given user ID) that follow the given activity
- /account_information
    - GET: returns information about the current account/user
- /edit_info
    - POST: updates account information for the current user

- /account_info_authentification
    - POST: checks whether credentials match the current user
- /join_group
    - POST: joins a given group
    - GET: returns a list of groups, users, and user membership data (debug information)
- /view_group/<group_id>
    - GET: returns information about a group for display on the View Groups page
- /contact_us
    - POST: send a communication to the developers
- /add_friend/<friend_id>
    - POST: add the given user as a friend to the current user
- /remove_friend/<friend_id>
    - DELETE: remove the given user from the current user’s friend list
- /friend_list/<user_id>
    - GET: returns the user’s friend list
- /kick_user/<user_id>/<group_id>
    - DELETE: removes the given user from the given group, if the current user is the group’s creator
- /is_friend/<user_id>
    - GET: checks whether the given user is the current user’s friend
- /profile_info/<user_id>
    - GET: returns profile data for a user, to display in their Profile page
- /leave_group/<group_id>
    - DELETE: have the current user leave the given group