OWL INTERNAL API
FOR APPLICATION-SERVER COMMUNICATION

1. User events
   1.1. Upon user sign-up
      1.1.1. Send HTTP post request to [EC2_DNS]/post/newuser
      1.1.2. JSON fields

uid	User ID generated by Firebase
fname	First name
lname	Last name
cyear	Class year (e.g. 2020, graduate)
netid	Netid
email	Email

      1.1.3. Example: {uid: “a1b2c3d4e5f6g7h8i9j1k2l3m4n5”, fname: “John”, lname: “Smith”, cyear: “2020”, netid: “jsmith”, email: “jsmith@princeton.edu”}
      1.1.4. Flask handler function: postnewuser()

   1.2. Upon user edit details
      1.2.1. Send HTTP post request to [EC2_DNS]/edit/existinguser
      1.2.2. JSON fields

uid	User ID generated by Firebase
fname	First name
lname	Last name
cyear	Class year (e.g. 2020, graduate)
netid	Netid
email	Email

      1.2.3. Example: {uid: “a1b2c3d4e5f6g7h8i9j1k2l3m4n5”, fname: “John”, lname: “Smith”, cyear: “2020”, netid: “jsmith”, email: “jsmith@princeton.edu”}
      1.2.4. Flask handler function: editexistinguser()

   1.3. Upon user preference selection
      1.3.1. Send HTTP post request to [EC2_DNS]/post/prefs
      1.3.2. JSON fields

deviceid		Device ID of sender
userid		UserID of sender
tags	location	List of location-related preferences
	category	List of category-related preferences
netid		Netid
email		Email

1.3.3. Example: {deviceid: "a1b2c3d4-12ab-1234-a123-a1b2c3d4e5f6", userid: "a1b2c3d4e5f6g7h8i9j1k2l3m4n5", tags: {location: {rocky: "false", mathey: "false", wilson: "false", butler: "true", whitman: "false", forbes: "true"}, category: {freefood: "false", brokenfacility: "true", recruiting: "false", studybreak: "false", movie: "true", busy: "false", firesafety: "false"}}}

1.3.4. Flask handler function: postprefs()

1.4. Upon event creation

1.4.1. Send HTTP post request to [EC2_DNS]/post/newevent

1.4.2. JSON fields

latitude	Latitude of event
longitude	Longitude of event
title	Title
description	Description
cat	Numeric identifier of type of event
oid	Owner ID (UserID that created event)
netid	Netid of owner
stime	Event creation time in UTC milliseconds
dur	Duration of event in minutes

1.4.3. Example: {latitude: 40.347379, longitude: -74.661435, title: “mathey study break”, description: “bubble tea”, cat: “1”, oid: “a1b2c3d4e5f6g7h8i9j1k2l3m4n5”, netid: “jsmith”, stime: “1493961244839”, dur: 60}

1.4.4. Flask handler function: postnewevent()

1.5. Upon user login

1.5.1. Send HTTP post request to [EC2_DNS]/post/prefs

1.5.2. Purpose is to register user and notification preferences on OneSignal
    1.5.2.1. Changes “active_status” in user tags on OneSignal to “true”
    1.5.2.2. Prevents mismatching issues related to different users logging in from same device
1.5.3. JSON: same as in section “Upon user preference selection”
1.5.4. Flask handler function: postprefs()

1.6. Upon user logout
1.6.1. Send HTTP post request to [EC2_DNS]/logout
1.6.2. JSON fields

deviceid	Device ID of sender

1.6.3. Example: {deviceid: "a1b2c3d4-12ab-1234-a123-a1b2c3d4e5f6"}
1.6.4. Flask handler function: loguserout()

1.7. Upon user muting notifications
1.7.1. Send HTTP post request to [EC2_DNS]/muteall
1.7.2. JSON fields

deviceid	Device ID of sender

1.7.3. Example: {deviceid: "a1b2c3d4-12ab-1234-a123-a1b2c3d4e5f6"}
1.7.4. Flask handler function: mutenotifs()

1.8. Upon user unmuting notifications
1.8.1. Send HTTP post request to [EC2_DNS]/unmuteall
1.8.2. JSON fields

deviceid	Device ID of sender

1.8.3. Example: {deviceid: "a1b2c3d4-12ab-1234-a123-a1b2c3d4e5f6"}
1.8.4. Flask handler function: unmutenotifs()

1.9. Upon user event-feedback submission
1.9.1. Send HTTP post request to [EC2_DNS]/vote
1.9.2. JSON fields

eventid	ID of affected event
upvotechange	1 if user upvoted, -1 if user undid upvote, 0 if upvote unaffected
downvotechange	1 if user downvoted, -1 if user undid downvote, 0 if downvote unaffected

1.9.3. Example: {eventid: “2f3fc9dd-902b-4c2c-9a26-4b5f4da3b3c3”, upvotechange: 1, downvotechange: 0}
1.9.4. Flask handler function: voteevent()

1.10. Upon user event deletion

  1.10.1. Send HTTP post request to [EC2_DNS]/post/deleteevent

  1.10.2. JSON fields

eventid	ID of affected event

  1.10.3. Example: {eventid: “2f3fc9dd-902b-4c2c-9a26-4b5f4da3b3c3”}

  1.10.4. Flask handler function: deleteevent()

2. User application background events

 2.1. Get all active events

  2.1.1. Send HTTP get request to [EC2_DNS]/get/allactive

  2.1.2. Flask app returns list of all active events

  2.1.3. Example: {[eventid: “2f3fc9dd-902b-4c2c-9a26-4b5f4da3b3c3”, latitude: 40.347379, longitude: -74.661435, title: “mathey study break”, description: “bubble tea”, category: “1”, ownerid: “a1b2c3d4e5f6g7h8i9j1k2l3m4n5”, netid: “jsmith”, starttime: “1493961244839”, duration: 60]}

  2.1.4. Flask handler function: getallactiveevents()

 2.2. Get user detail

  2.2.1. Send HTTP get request to [EC2_DNS]/get/userinfo/?desired=[field]&uid=[uid]

  2.2.2. Flask app returns the desired field of the user with userid uid

  2.2.3. Example request: [EC2_DNS]/get/userinfo/?desired=fname&uid=a1b2c3d4e5f6g7h8i9j1k2l3m4n5

  2.2.4. Flask handler function: getuserinfo()

 2.3. Get event detail

  2.3.1. Send HTTP get request to [EC2_DNS]/get/eventinfo/?desired=[field]&eid=[eid]

  2.3.2. Flask app returns the desired field of the event with eventid eid

  2.3.3. Example request: [EC2_DNS]/get/eventinfo/?desired=title&eid=2f3fc9dd-902b-4c2c-9a26-4b5f4da3b3c3

  2.3.4. Flask handler function: geteventinfo()