List of backend API

List of methods for iOS app

Return Type:

var res200 = {

status: 200,

statusText: "OK"

};

var res404 = {

status: 404,

statusText: "Not Found"

};

var res500 = {

status: 500,

statusText: "Firebase Error"

};

API:

1. SignIn and create new preference

Input: User data that we can get from Facebook.

Return: User | Error.

Note: you should tell me how to upload files (images or video)

API: POST

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/users>

DESC:

PARAMETER:

newuser: new user data by json

Return: if success, userid, otherwise res500

1. GetCurrentUser (I need to get currently logged in user. Looks like I need to send you some input data so you will be able to check whether the user account exists. According to the fact that we are signing in through the Facebook, should I send you personal facebook user id?)

Input: ?

Return: User | Error.

API: GET

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/users/10154390471048223>

DESC: 10154390471048223:fbid

PARAMETER:no

Return: if success, user data, otherwise res404

1. UploadFilesForUser (It may be images or videos)

Input: Array of files (one or more), user Id

Return: User | Error

/upload/:userid/:type/:oldurl

API: POST

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/upload/:userid/:type/>

DESC:

userid : user id to be replace

type : main or other (whether mainProfileImage or otherProfileImages)

PARAMETER: 'uploadFile' : new file(image or video)

thumbnailImage : thumbnail image for video

oldurl : old url to be replaced( if oldurl is empty, upload new image, otherwise replace)

Return: if success, user data, otherwise res404

1. GetPreferencesByUserId or GetCurrentPreferences

Input: user id.

Return: Preferences | Error.

API: GET

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/preferences/10154390471048223>

DESC: 10154390471048223:userid

PARAMETER:no

Return: if success, preference data, otherwise res404

1. UpdateUserPreferencesInfo

Input: newPreferences, user id

Return: Preferences | Error.

API: PUT

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/preferences/10154390471048223>

DESC: 10154390471048223:userid

PARAMETER:

newpref : new preference data

Return: if success, res200, otherwise res404

1. UpdateUserInfo

Input: newUser

Return: User | Error

API: PUT

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/users/10154390471048223>

DESC: 10154390471048223:userid

PARAMETER:

newuser: new user data

Return: if success, res200, otherwise res404

1. GetMomentsByUsedId (by default they are sorted 'newest')

Input: user Id(обязательно), sorting flag (newest/most popular), filter flag (interest id).

Return: [Moment].

API: GET

[http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/moments/10154390471048223 /sort/filter](http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/moments/10154390471048223%20/sort/filter)

DESC: 10154390471048223:userid

sort: updateAt or popular

filter: string for momentsPassion ( “all” for all)

PARAMETER:

Return: if success, [moment with user], otherwise res404

1. GetAllMoments (we can combine this and the previous method in one)

Input: sorting flag (newest/most popular), filter flag (interest id)

Return: [Moment]

API: GET

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/moments/sort/filter>

DESC:

sort: updateAt or popular

filter: string for momentsPassion ( “all” for all)

PARAMETER:

Return: if success, [moment with user], otherwise res404

1. GetMatchedMomentsByUserId

Input: user Id, sorting flag (newest/most popular), filter flag (interest id)

Return: [Moment]

API: GET

[http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/matchedmoments/10154390471048223/sort/filter](http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/matchedmoments/userid/sort/filter)

DESC: 10154390471048223:userid

sort: updateAt or popular

filter: string for momentsPassion ( “all” for all)

PARAMETER:

Return: if success, [moment with user], otherwise res404

1. CreateMoment

Input: - Info about new moment (please take a look on the «Image upload.png» screen)

Return: Moment | Error.

API: PUT

[http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/moments](http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/moments/sort/filter)

DESC:

PARAMETER:

newmoment : new moment data

Return: if success, new moment id, otherwise res404

1. GetAllInterests (It means «travel», «foodie», etc.)

Input: -

Return: [Interest].

API: GET

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/passions>

DESC:

PARAMETER:

Return: if success, [interest], otherwise res404

1. GetLikersForMomentByMomentId

Input: moment id, user id

Return: [User]

API: GET

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/likemoments/users/:momentid>/:userid

DESC:

PARAMETER:

momentid : moment object id to search

userid : current user id

Return: if success, [User], otherwise res404

[User] includes status between userid and liker

1. ReportMoment

Input: moment id, user id (who is reporting)

Return: Success | Error

API: POST

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/reportmoment/:momentid/:userid>

DESC:

momentid: moment id

userid: userid

PARAMETER:

Return: if success, res200, otherwise res500

1. GetNotificationsForUserId (Please take a look on screen «Notification.png». There are two types of notifications «people» and «likes». You need to tell me how this method will work – whether return all notification and I will split them by type, or add some type flag in request.)

Input: userId

Return: [Notification].

API: GET

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/notifications/:userid>

DESC:

userid : userid

PARAMETER:

Return: if success, [notification], otherwise res404

1. GetUser (Please add the status between these users (whether one of them has already accepted/declined the other one, or they are matched)

Also is it possible to receive an information about mutual friends?

Input: user id (the person who is searching), user Id (the person who has been searched)

Return: User.

API: GET

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/match/:userid1/:userid2>

DESC:

Userid1 : the person who is searching

Userid2 : the person who has been searched

PARAMETER:

Return: if success, status code, otherwise res404

Return code:

Both of them didn’t do anything 1

User1 rejected user2, user2 didn’t do anything 2

User1 approved user2, user2 didn’t do anything 3

User1 didn’t do anyhing, user2 rejected user1 4

User1 reejcted user2, user2 rejected user1 5

User1 approved user2, user2 rejected user1 6

User1 didn’t do anything, user2 approved user1 7

User1 rejected user2, user2 approved user1 8

User1 approved user2, user2 approved user1 9

1. LikeMoment

Input: user Id, moment Id.

Return: Success | Error

API: PUT

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/likemoments/:userid/:momentid>

DESC:

userid : user id

momentid : moment id

PARAMETER:

Return: if success, moment, otherwise res404

1. unlikeMoment

Input: user Id, moment Id.

Return: Success | Error

API: DELETE

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/likemoments/:userid/:momentid>

DESC:

userid : user id

momentid : moment id

PARAMETER:

Return: if success, moment, otherwise res404

1. DeleteMoment

Input: user Id, moment Id.

Return: Moment (the one which was deleted) | Error.

API: DELETE

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/moments/:userid/:momentid>

DESC:

userid : user id

momentid : moment id

PARAMETER:

Return: if success, res200, otherwise res404

1. ReportUser

Input: reporter user id, reported user id

Return: Success | Error

API: POST

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/reportuser/:userid1/:userid2>

DESC:

userid1 : reporter user id

userid2 : reported user id

PARAMETER:

Return: if success, res200, otherwise res404

1. DownloadEvent ("downloadEvent") (This method is not related to Events functionality) (Please take a look in the previous realization or ask Josh about this method because I can see such method is used in the app, but I am not sure why, so perhaps we need it or not)

Input: user id, times (Int).

Return: Success | Error

API: POST

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/downloadevent/:userid1/:times>

DESC:

userid1 : reporter user id

times : times

PARAMETER:

Return: if success, res200, otherwise res500

1. ApproveMatch

Input: approver userId, user id.

Return: Success | Error

API: POST

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/match/:userid1/:userid2>

DESC:

userid1 : approver user id

userid2 : approved user id

PARAMETER:

Return: if success, res200, otherwise res500

1. Decline match

Input: decliner user id, user id.

Return: Success | Error

API: DELETE

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/match/:userid1/:userid2>

DESC:

userid1 : decliner user id

userid2 : declined user id

PARAMETER:

Return: if success, res200, otherwise res500

1. GetUsersForMatch (name in the previous realization "possibleMatches") (I'm not sure but I think this method should get also location, but please ask Josh how it should work)

Input: userId

Return: [User].

API: GET

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/approveuserformatch/:userid>

DESC:

userid : user id

PARAMETER:

Return: if success, [User], otherwise res404

1. GetMatchesForUser (name in the previous realization " matches") (Please ask Josh how it should work)

Input: userId

Return: [User].

API: GET

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/approvematchforuser/:userid>

DESC:

userid : user id

PARAMETER:

Return: if success, [User], otherwise res404

1. GetCountriesForUser

Input: user id

Return: [Country].

API: GET

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/countries/10154390471048223>

DESC:

10154390471048223 : userid

PARAMETER: no

Return: if success, [country], otherwise res404

1. AddUserCountriesForUser

Input: user id, country code.

Return: Country code .

API: PUT

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/countries/10154390471048223>

DESC:

10154390471048223 : userid

PARAMETER:

country : country code (i.e. SG for Singapore, AU for Australia etc)

Return: if success, [country], otherwise res404

1. DeleteCountryForUser

Input: user id, country code.

Return: Country code.

API: DELETE

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/countries/10154390471048223>

DESC:

10154390471048223 : userid

PARAMETER:

country : country code (i.e. SG for Singapore, AU for Australia etc)

Return: if success, [country], otherwise res404

1. GetEventsForUser (This functionality is not clear so we need to talk about it all together)

Input: userId (perhaps you need to get also user's location?)

Return: [Events]

Input: user id

Return: [event]

API: GET

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/events/00rwER5tfB>

DESC:

00rwER5tfB : user id

PARAMETER:

Return: if success, [event], otherwise []

1. GetMomentById

Input: user id, country code.

Return: Moment

API: GET

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/moments/00rwER5tfB>

DESC:

00rwER5tfB : moment id

PARAMETER:

Return: if success, moment , otherwise res404

1. updateMoment

Input: moment id

Return: success|error

API: PUT

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/moments/00rwER5tfB>

DESC:

00rwER5tfB : moment id

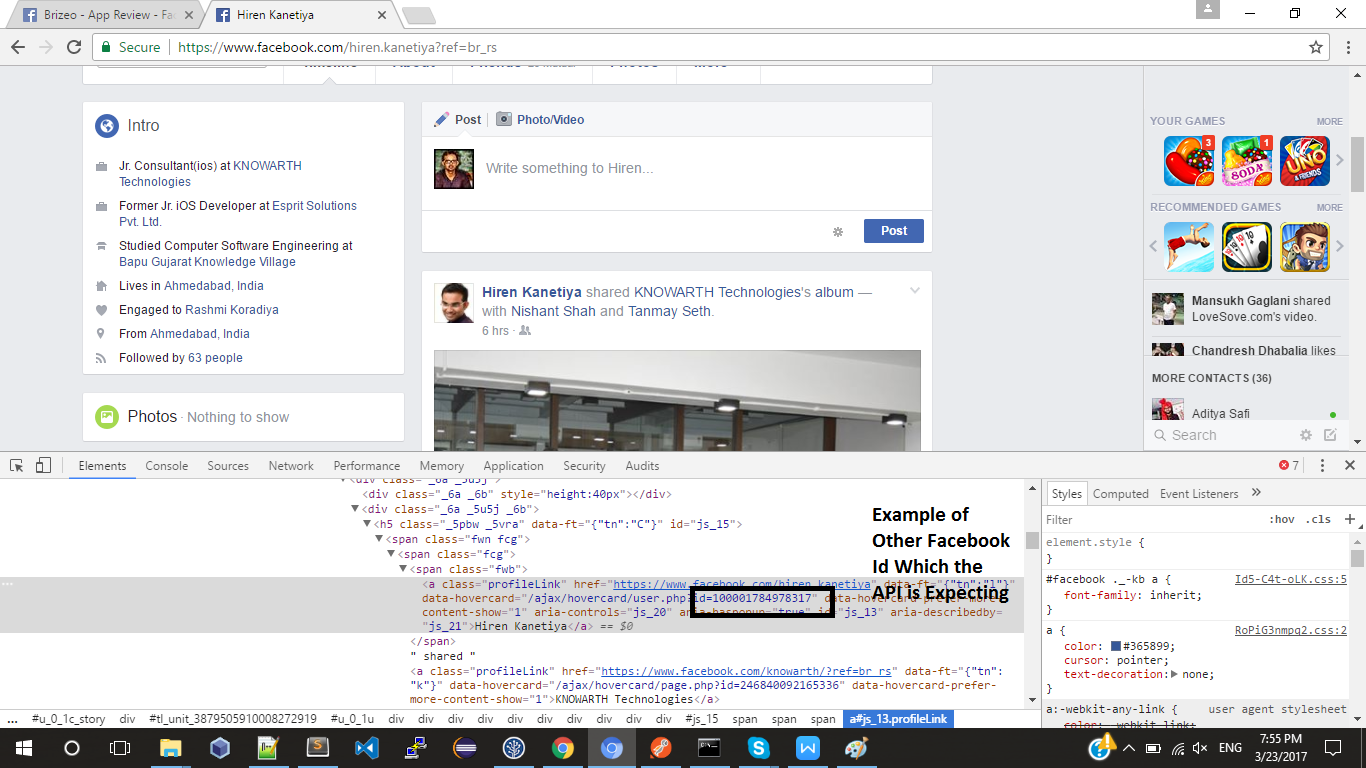
PARAMETER: newmoment: new moment information

Return: if success, 200, otherwise 404

1. GetMutualFriends

Input : Access Token Of Signed In User & Facebook User Id of the person whose mutual friends I need to see.

Example : if I want to see Mutual friends of “ABC Person” this is the ID which the API Is Expecting.



Return : List of Mutual Friends.

API: GET

[site name + /m](http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/events)utual\_friends/:otheruserId/:accessToken

Example Link :

mutual\_friends/100001784978317/EAAVVdMVpvTEBAB7Ceiqsqe3qO4yZCWahZC7PGsircm2pOXlwxeN0JryL7J63HoopoSMR3d1BjZBVuLLhdBk3mqV7lVDYv9VJwvkcs9DUyfoF2Fn5VM36tdzzDtphyaxa5ztU8NBEV51xRFPvD4w05zi44EaC0hMDYpJoYZCZBp5h8f9jMNbT7VPhbZBvJuFMMZD

DESC: It returns list of mutu

Return: if success, returns list of mutual friends.

Sample Response :

[

{

"name": "Ketan Savaliya",

"token": "QVNrMEJGdW53WjN0Tk5JZATNpS2ZAOOEgzdGhpWklaU1J5dUhEVF9kZAXp6c0NzTERnQ0xEN0F5SWJMSDBTMk0teTNaVUh6ZAk1BUG9XY18teUNtQXJ5eEUxY0U4Nk05R1l0Sk5jUzk3OFRRSUIweGcZD",

"picture": {

"data": {

"is\_silhouette": false,

"url": "https://scontent.xx.fbcdn.net/v/t1.0-1/c8.0.50.50/p50x50/16681745\_1358136457594760\_4753566363422856613\_n.jpg?oh=78d150315c0d9419bd1d694e613ca3ff&oe=59670AFA"

}

}

}

….

]

1. Save new event data

Input: new event data

Return: event id | error

API: POST

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/events>

DESC:

PARAMETER: newevent: new event data

Return: if success, new event id, otherwise 404

1. Update notification

Input : new notification data

Return : success | error

API: PUT

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/notifications/00rwER5tfB>

DESC:

00rwER5tfB : notification id

PARAMETER: newnotification: new notification data

Return: if success, 200, otherwise 404

1. Get all events

Input : sort type and location

Return : [event] | error

API: PUT

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/allevents/:sort>

DESC:

sort : “nearest” or “popular”

PARAMETER:

lat : search latitude

lon : search longitude

Return: if success, searched events, otherwise 404

1. Get valid fb users

Input : [fbid]

Return : [valid users] | error

API: POST

<http://ec2-54-244-190-99.us-west-2.compute.amazonaws.com:3000/fbusers>

DESC:

PARAMETER:

fbids : facebook user ids

Return: if success, valid users, otherwise empty array

1. Get Events By User

Description : All the events user is attending or interested in.

API : PUT

<http://whatever-address/events-by-user/:userId>/:sort

Example :http://localhost:3000/events-by-user/-Kgs67ymCbrBfkE01TAA/distance

lat : 49.824481263791

lon":" 24.038967132569"

PARAMETERs :

lat : search latitude

lon : search longitude

Returns: Output of events user is attending or interested in based on filter parameter.

Example :

{

"eventsUserInvolved": [],

"status": 200

}

Note : if I filter by distance, then it will have 1 more parameter distance.