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/usres>

DESC:

PARAMETER:

newuser: new user data by json

Return: if success, res200, 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

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], 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], 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/sort/filter>

DESC: 10154390471048223:userid

sort: updateAt or popular

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

PARAMETER:

Return: if success, [moment], 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, [moment], 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/interests>

DESC:

PARAMETER:

Return: if success, [interest], otherwise res404

1. GetLikersForMomentByMomentId

Input: moment id

Return: [User]

API: GET

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

DESC:

PARAMETER:

momentid : moment object id to search

Return: if success, [User], otherwise res404

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, [User], otherwise res404

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 has been searched), user id (the person who is searching).

Return: User.

API: GET

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

DESC:

userid1 : the person who has been searched

userid2 : the person who is searching

PARAMETER:

Return: if success, User, otherwise res404

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, res200, 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, res200, 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/:fbid1/: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 : user id

userid2 : 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 : user id

userid2 : 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: DELETE

<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: DELETE

<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]