**The Whoot API V2**

**Last updated: May 2, 2012**

**Notes**

- The production website is at [thewhoot.com](http://www.thewhoot.com), the staging website is at staging.thewhoot.com. Use the staging website for testing.

- All URLS (except for user search) are namespaced to /api/v2/{endpoint}

- All responses are JSON

- Always include the parameter ‘auth\_token’ to authenticate the user

- A response code of 200 indicates the request was successful

- Form submissions with errors will respond with a 422 error code. The errors will be in the following format:

{

‘status’:’error’,

‘errors’:

[

{field}:’error’,

{field}:’error’

]

}

- All other errors will include the following information

{

‘status’:’error’,

‘message’:’’,

‘code’:123,

}

- \* indicates a required parameter

- We’re using the google maps API for the address/venue lookup field.

- Facebook App IDS:

Production – 224248554281682

Staging - 120786798012675

**POST ‘generate\_token’**

*This endpoint will generate a whoot user token that you can use on future requests to authenticate the user. If the user is not already a member of the whoot, it will create an account for them.* ***The app must first get a facebook session token for the user tied to our facebook app.***

**Params:**

**\***token - The Facebook session token

**Response:**

token – The Whoot user token

**GET ‘posts/feed’**

*This endpoint will return an array of normal posts for the current user. It returns data representing the users main feed.*

**POST ‘posts’**

*This endpoint will create a new normal post for the user.*

**Params:**

**\***night\_type– (string) working | low\_in | low\_out | big\_out

tag – (string) 40 char max

address\_original – (string) What was originally entered into the venue box

venue\_address – (string) Venue address returned from google

venue\_name – (string) Venue name returned from google

suggestions (boolean) Whether the user is open for suggestions tonight

**GET ‘users/undecided’**

*This endpoint returns a list of undecided users that the current user is following.*

**POST ‘comments’**

*This endpoint will create a new comment on a post for the user.*

**Params:**

**\*** content – (string) The text content of the comment

\* post\_id – (string) The post id this comment is for

**POST ‘posts/loop\_ins**

*This endpoint will loop the user into a post.*

**Params:**

**\***id – the post id

**POST ‘users/follows’**

*This endpoint will cause the current user to follow the target user.*

**Params:**

**\***id – the target user id (the user this user should follow)

**DELETE ‘users/follows’**

*This endpoint will cause the current user to stop following the target user.*

**Params:**

**\***id – the target user id (the user this user should follow)

**GET ‘me’**

*This endpoint returns info for the current user.*

**GET ‘users/:id’**

*This endpoint returns an array of users that the given user is following.*

**Params:**

**\***id – the target users id

**GET ‘users/:id/following\_users’**

*This endpoint returns an array of users that the given user is following.*

**Params:**

**\***id – the target users id

**GET ‘users/:id/followers’**

*This endpoint returns an array of users following the given user.*

**Params:**

**\***id – the target users id

**POST ‘ping’**

*This endpoint will cause the current user to ping the target user.*

**Params:**

**\***id – the target user id (the user the current user should ping)

**PUT ‘users’**

*This endpoint will update attributes of the current user*

**Params:**

email\_comment – (Boolean) Whether to notify this user when somebody comments on their post

email\_ping – (Boolean) Whether to notify this user when somebody pings them

email\_follow – (Boolean) Whether to notify this user when somebody follows them

email\_daily – (Boolean) Whether to send this user a daily digest of notifications

unread\_notification\_count – (integer) The number of unread notifications the user currently has

**GET ‘users/blocked’**

*This endpoint will return a list of currently blocked users*

**POST ‘users/blocked’**

*This endpoint will block a user*

**Params:**

**\***id – the target user id (the user the current user should block)

**DELETE ‘users/blocked’**

*This endpoint will unblock a user*

**Params:**

**\***id – the target user id (the user the current user should unblock)

**GET ‘users/notifications**

*This endpoint will return a list of user notifications. See the PUT ‘users’ endpoint to clear the unread notification count.*

**GET ‘analytics’**

*This data returns nightly analytics for the current user*

**GET ‘soul-data/search’**

*This endpoint will cause the current user to follow the target user.*

**Params:**

\*term: (string) the search term

\*limit: (int) limit number of results

\*types[]: ‘user’ AND/OR user\_id

If the user\_id is present the search service will return only users that the user\_id is following that match the term. If just ‘user’ is present as the type the search will return users matching the term. If both a user\_id and ‘user’ is present it will return two sets of results. One set will be users the user\_id is following that match the term and the other will be other users that match the term. There will be no duplicates.

*here is an example request:*

staging.thewhoot.com/soul-data/search?term=marc&limit=10&types[]=4ec1480acddc7f4de0000011&types[]=user

What this will do is return a set of results for users the 4ec1480acddc7f4de0000011 user is following that match ‘marc’ and general users that match ‘marc’.