Account management

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description | method | Uri | header | body | Code 200 response |
| Signup a new user | POST | /users |  | {email, password} | Header:  x-auth  body:  user object |
| Login | POST | /users/login |  | {email, password} | Header:  x-auth  body:  user object |
| logout | DELETE | /users/me/token | x-auth |  |  |
| Get user’s profile | GET | /users/me | x-auth |  | Header:  x-auth  body:  user object |
| Get socket connection ticket | GET | /chatConnections | x-auth |  | Header:  x-auth  {chatConnection<url for websocket>} |
| User requests to close an existing conneciton | DELETE | /chatConnecitons |  |  |  |

Parking space

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description | method | Uri | header | body | Code 200 response |
| User cleared a parking space | POST | /parkingSpaces |  | {latitude, longitude} |  |
| User took parking space by coordinates | DELETE | /parkingSpaces |  | {latitude, longitude, id}//coordinates or id |  |
| user creates a search for a nearby parking spaces – sends his coordinates  And receives | POST | /parkingSpaces/searches |  | {latitude, longitude, distance(meters)} | {searchID} |
| user creates a search for a nearby parking spaces – sends an address | POST | /parkingSpaces/searches |  | {address, distance(meters)} | {searchID} |
| User retrieves results for a search | GET | /parkingSpaces/searches/{id} |  |  | {latitude, longitude, distance, timestamp, results:[]} |
|  |  |  |  |  |  |

**Community chat**:

To create a websocket connection for chat the client should:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description | method | Uri | header | body | Code 200 response |
| Get socket connection ticket | GET | /chatConnections | x-auth |  | Header:  x-auth  {chatConnection<url for websocket>} |
| User requests to close an existing connection | DELETE | /chatConnecitons | x-auth |  | Header:  x-auth |

1. Send GET request to /chatConnecitons including the jwt in x-auth
2. If a connection exists it will return 400
   1. To handle this scenario the client can send DELETE to /chatConnections – it will close the existing connection allowing it to create the new connection
3. the response will include a url to be used for creating the websocket connection

Notes:

* currently one chat room and should evolve into several chats
* first the chat will be over websocket while the client will keep using the above REST API in parallel
* in the future the client will hopefully consume the API via the websocket avoiding 2 connections per client scenario
* only one chat connection is permitted per client. A new request will kill existing connection
* once the client logout the websocket connection will be closed

Questions:

1. Should the users be registered? What privileges will they get once registered?
2. What about the accuracy of the parking space? User A published available parking on {long, lat} and user B reported that he took it with the location of .
   * A more direct question – how do we catalog the parking spaces? By coordinates? Or should there be additional identifying data?
   * What if we have more than one available parking space in the same location?

Future tasks(gal):

* When using a string address
* Sort parking spaces results by distance
* Currently all searches are saved as documents in the DB – optimize so that similar searches will not be saved
* set and implement TTL mechanism for the search results
* Implement “//future todo implement”

Distant future tasks(gal):

* Clear unused nodejs packages