Project 2: Microblog Microservices REST API Documentation

Service: Users

createUser(username, email, password)
Resource URL Endpoint: /users/create
HTTP Method: POST
HTTP Status Codes:

- Success: 200 OK
- Failure: 409 CONFLICT

Request JSON Data Format:

```
{
    'username' : <String>
    'email' : <String>
    'password' : <String>
}
```

Response JSON Data Format:

```
{
    'success' : 'User account successfully created'
}
```

Description: Creates a new user with the specified username, email and password. If the username is already in the database, a 409 CONFLICT will be returned. If successful, the new user will be stored in the database to be used to authenticate later. The specified password is hashed before being stored in the database.

authenticateUser(username, hashed_password)
Resource URL Endpoint: /users/authenticate
HTTP Method: GET
HTTP Status Codes:

- Success: 200 OK
- Failure: 403 FORBIDDEN

Response JSON Data Format:

```
{
    'success' : 'User account successfully authenticated'
}
```

Description: Authenticates the user by comparing the specified username and hashed password with the entries in the database. If there is a discrepancy, 403 FORBIDDEN is returned.

<u>addFollower(username, usernameToFollow)</u>
Resource URL Endpoint: /users/follow
HTTP Method: POST
HTTP Status Codes:

- Success: 200 OK
- Failure:
    - 400 BAD REQUEST (user does not exist)
    - 409 CONFLICT

Request JSON Data Format:

```
{
        'followee' : <String>
        'follower' : <String>
}
```

Response JSON Data Format:

```
{
        'success' : 'User <follower> has started following <followee>'
}
```

Description: Adds a follower, followee pair to the database by storing the usernames of each as one object in the userfollowers table. If either specified username does not exist in the database, a 400 BAD REQUEST is returned.

<u>removeFollower(username, usernameToRemove)</u>
Resource URL Endpoint: /users/unfollow
HTTP Method: DELETE
HTTP Status Codes:

- Success: 200 OK
- Failure: 400 BAD REQUEST

Request JSON Data Format:

```
{
        'followee' : <String>
        'follower' : <String>
}
```

Response JSON Data Format:

```
{
        'success' : 'User <follower> has stopped following <followee>'
}
```

Description: Removes a follower, followee pair from the userfollowers table in the database. If the username does not exist in the database, or is not following the specified followee according to the database, a 400 BAD REQUEST is returned.

Service: Timelines

getUserTimeline(username)
Resource URL Endpoint: /userTimeline
HTTP Method: GET
HTTP Status Codes:

- Success: 200 OK
- Failure: 400 BAD REQUEST (user does not exist)

Response JSON Data Format:

```
{
    {
        'author' : <string>
        'post' : <string>
        'posttimestamp' : <datetime>
    }
    ...
    {
        'author' : <string>
        'post' : <string>
        'posttimestamp' : <datetime>
    }
}
```

Description: Get a user's timeline which includes the 25 most recent tweets that the user has posted. Tweets are sorted in order from most recent to least recent. If the requested user timeline is for a user that does not exist in the database, a 400 BAD REQUEST.

getPublicTimeline()
Resource URL Endpoint: /timeline
HTTP Method: GET
HTTP Status Codes:

- Success: 200 OK

Response JSON Data Format:

```
{
    {
        'author' : <string>
        'post' : <string>
        'posttimestamp' : <datetime>
    }
```

```
        ...
        {
            'author' : <string>
            'post' : <string>
            'posttimestamp' : <datetime>
        }
    }
```

Description: Get the public timeline which includes the 25 most recent tweets from all users. Tweets are sorted in order from most recent to least recent.

<u>getHomeTimeline(username)</u>
Resource URL Endpoint: /home
HTTP Method: GET
HTTP Status Codes:

- Success: 200 OK
- Failure: 400 BAD REQUEST (user does not exist)

Response JSON Data Format:

```
    {
        {
            'author' : <string>
            'post' : <string>
            'posttimestamp' : <datetime>
        }
        ...
        {
            'author' : <string>
            'post' : <string>
            'posttimestamp' : <datetime>
        }
    }
```

Description: Get the home timeline which includes the 25 most recent tweets from all of the users that a specified user follows. Tweets are sorted in order from most recent to least recent. If the requested home timeline is for a user that does not exist in the database, a 400 BAD REQUEST will be returned.

<u>postTweet(username, text)</u>
Resource URL Endpoint: /tweet/create
HTTP Method: POST
HTTP Status Codes:

- Success: 200 OK

- Failure: 400 BAD REQUEST (user does not exist)

Request JSON Data Format:

```
{
    'user' : <String>
    'text' : <String>
}
```

Response JSON Data Format:

```
{
    'success' : 'Tweet successfully posted'
}
```

Description: Post a tweet under the specified username. Tweets include an author, the user who posted the tweet, text, the body of the tweet, and a timestamp which is generated by the function. The tweet is stored in the database to be retrieved later. If the user that is specified does not exist in the database, a 400 BAD REQUEST will be returned.