Kazander Antonio
Daniel Sollis

Project 1: Minitwit API Documentation

get_users():

URL: /api/users

Type: GET

Description: Get all the users for minitwit.

Request:

No request auth headers or body necessary

Reply Json Example:

```
{
   "users": [
      {
         "email": "foo@bar.com",
         "pw_hash": "pbkdf2:sha256:50000$vc29t3be$733a48d6f8327f52aee08d9f6c465fdabba28feedd519f6d567a08a8ad1fb099",
         "user_id": 1,
         "username": "Daniel"
      },
      {
         "email": "bar@foo.com",
         "pw_hash":
"pbkdf2:sha256:50000$UiiOD3Ph$2e8b8bf3e687b9c864c6927d77ed534b40e3a3a7bd18dc340f178caa84c32c29",
         "user_id": 2,
         "username": "Sollis"
      },
      {
         "email": "foo@foo.com",
         "pw_hash": "pbkdf2:sha256:50000$VX8nBZtf$50649a866afe2289607b32db05be486ec6d8564aec77ff71cc1c9e816fb7cf12",
         "user_id": 3,
         "username": "Kaz"
      },
      {
         "email": "bar@bar.com",
         "pw_hash":
"pbkdf2:sha256:50000$i6clB34o$524077f24e470c95b53e0646965b1d5c80e8360eca9a1977c84f25c10bc907b5",
         "user_id": 4,
         "username": "Antonio"
      }
   ]
}
```

get_public():

URL: /api/public

Type: GET

Description: Get the public timeline for minitwit.

Request:

No request auth headers or body necessary

Reply Json Example:

```
{
   "public timeline": [
      {
         "author_id": 1,
    "message_id": 1,
    "pub_date": 6,
    "text": "I",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 2,
    "pub_date": 5,
    "text": "can",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 3,
    "pub_date": 4,
    "text": "code",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 4,
    "pub_date": 3,
    "text": "Im",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 5,
    "pub_date": 2,
    "text": "not a",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 6,
    "pub_date": 1,
    "text": "moron!",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 7,
    "pub_date": 0,
    "text": "oy",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 8,
    "pub_date": 0,
    "text": "yo",
    "username": "Daniel"
},
{
    "author_id": 2,
    "message_id": 9,
            "pub_date": 0,
            "text": "hey",
            "username": "Sollis"
        },
        {
            "author_id": 3,
            "message_id": 10,
            "pub_date": 0,
            "text": "sup",
            "username": "Kaz"
        },
        {
            "author_id": 4,
            "message_id": 11,
            "pub_date": 0,
            "text": "!",
            "username": "Antonio"
        }
    ]
}
```

users_timeline():
URL: /api/users/<username>/timeline
Type: GET
Description: Get the timeline for a specific user
Request:
No request auth headers or body necessary
Reply Json Example:

```
{
    "Kaz's timeline": [
        {
            "author_id": 3,
            "message_id": 10,
            "pub_date": 0,
            "text": "sup"
        }
    ]
}
```

Failure (User doesn't exist) Json Example:

```
{
    "status code": "404"
}
```

add_user():
URL: /api/register
Type: GET
Description: Add a new user to minitwit. Authorization is required, as well as an email form in the body of the request.
Request:

Request Headers:

content-type:"multipart/form-data; boundary=--------------------------098828582845053721309142"

cache-control:"no-cache"

postman-token:"6efe58ce-ef98-46bb-adf5-5207008c27f1"

authorization:"Basic ZG9ubmE6bW9tbWE="

user-agent:"PostmanRuntime/7.1.1"

accept:"*/*"

host:"localhost:5000"

accept-encoding:"gzip, deflate"

content-length:175

Request Body:

email:"zomma@momma.com"

Success Reply Json Example:

```
{
  "message": "Success, user has been added."
}
```

Failure Reply (User already Exists) Json Example:

```
{
  "message": "That username is already taken. Please try a different username."
}
```

Failure Reply (No Email) Json Example:

```
{
  "message": "There was no email for the user in the request body. Please add the user's email in the 'email' form in the request
body"
}
```

get_followers():

URL: /api/users/<username>/followers

Type: GET

Description: Get the followers for a user.

Request:

No request auth headers or body necessary

Success Json Example:

```
{
  "followers": {
    "1": "Daniel"
  }
}
```

Failure (User doesn’t exist) Json Example:

```
{
  "status code": "404"
}
```

get_following():
URL: /api/users/<username>/following
Type: GET
Description: Get the list of followed users for a user.
Request:
No request auth headers or body necessary
Success Json Example:

```
{
   "following": {
      "1": "Daniel"
   }
}
```

Failure (User doesn't exist) Json Example:

```
{
   "status code": "404"
}
```

insert_message():
URL: /api/users/<username>/post
Type: POST
Description: Post a new message for the user. Requires proper authentication headers. Message has to go into request body.
Request:
Request Headers:

```
content-type:"multipart/form-data; boundary=--------------------------565557218844566474334623"
cache-control:"no-cache"
postman-token:"41fc81ee-409b-40ff-999b-9f93345ae707"
authorization:"Basic bW9tbWE6bW9tbWE="
user-agent:"PostmanRuntime/7.1.1"
accept:"*/*"
host:"localhost:5000"
accept-encoding:"gzip, deflate"
content-length:177
```

Request Body:

```
message:"WEEEEE HOOOOOOO"
```

Success Json Example:

```
{
   "message": "Success, you've made a post."
}
```

Failure (Not You) Json Example:

```
{
   "status code": "403 Forbidden: You cannot post to a user that isn't you"
}
```

Failure (No Message) Json Example:

```
{
   "Error": "There was no message in the request body. Please add what you would like to post under the 'message' form in the
request body"
}
```

Flask-BasicAuth:

On failed authentication Flask-BasicAuth replies with a 401: Unauthorized, though not in Json.

get_dash():

URL: /api/users/<username>/dashboard

Type: GET

Description: Get dashboard (Messages of all followed users) for a user. Requires proper authentication headers.

Request:

Request Headers:

content-type:"application/x-www-form-urlencoded"

cache-control:"no-cache"

postman-token:"223aa0df-f615-472b-92e8-e63284a0d9ea"

authorization:"Basic bW9tbWE6bW9tbWE="

user-agent:"PostmanRuntime/7.1.1"

accept:"*/*"

host:"localhost:5000"

accept-encoding:"gzip, deflate"

Success Json Example:

```
{
   "dashboard": [
      {
         "author_id": 5,
         "message_id": 12,
         "pub_date": 1519101071.357537,
         "text": "I EAT GRAPES",
         "username": "momma"
},
{
    "author_id": 1,
    "message_id": 1,
    "pub_date": 6,
    "text": "I",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 2,
    "pub_date": 5,
    "text": "can",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 3,
    "pub_date": 4,
    "text": "code",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 4,
    "pub_date": 3,
    "text": "Im",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 5,
    "pub_date": 2,
    "text": "not a",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 6,
    "pub_date": 1,
    "text": "moron!",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 7,
    "pub_date": 0,
    "text": "oy",
    "username": "Daniel"
},
{
    "author_id": 1,
    "message_id": 8,
    "pub_date": 0,
    "text": "yo",
    "username": "Daniel"
}
```

```
  ]
}
```

Failure (Not You) Json Example:

```
{
  "status code": "403 Forbidden: This dashboard doesn't belong to you"
}
```

Flask-BasicAuth:

On failed authentication Flask-BasicAuth replies with a 401: Unauthorized, though not in Json.

api_follow():

URL: /api/users/<username>/follow/<desired_followed_user>

Type: DELETE

Description: Allows you to follow another user. Requires proper authentication headers.

Request:

Request Headers:

```
content-type:"application/x-www-form-urlencoded"
cache-control:"no-cache"
postman-token:"bcd2392b-fece-4f4c-98a8-ec0e07f2ca0e"
authorization:"Basic bW9tbWE6bW9tbWE="
user-agent:"PostmanRuntime/7.1.1"
accept:"*/*"
host:"localhost:5000"
accept-encoding:"gzip, deflate"
content-length:""
```

Success Json Example:

```
{
  "message": "Success, You are now following Daniel"
}
```

Failure (Not You) Json Example:

```
{
  "status code": "403 Forbidden: You're trying to make someone who isn't you follow someone else."
}
```

Failure (Same User) Json Example:

```
{
  "Error": "You can’t follow yourself"
}
```

Failure (User doesn’t exist) Json Example:

```
{
   "status code": "404: User not found."
}
```

Flask-BasicAuth:
On failed authentication Flask-BasicAuth replies with a 401: Unauthorized, though not in Json.

api_unfollow():
URL: /api/users/<username>/unfollow/<desired_followed_user>
Type: DELETE
Description: Allows you to unfollow another user. Requires proper authentication headers.
Request:
Request Headers:

```
        content-type:"application/x-www-form-urlencoded"

        cache-control:"no-cache"

        postman-token:"bcd2392b-fece-4f4c-98a8-ec0e07f2ca0e"

        authorization:"Basic bW9tbWE6bW9tbWE="

        user-agent:"PostmanRuntime/7.1.1"

        accept:"*/*"

        host:"localhost:5000"

        accept-encoding:"gzip, deflate"

        content-length:""
```

Success Json Example:

```
{
   "message": "Success, you unfollowed Daniel"
}
```

Failure (Not You) Json Example:

```
{
   "status code": "403 Forbidden: You're trying to make someone who isn't you follow someone else."
}
```

Failure (Same User) Json Example:

```
{
   "Error": "You can’t unfollow yourself"
}
```

Failure (User doesn’t exist) Json Example:

```
{
   "status code": "404: User not found."
}
```

Flask-BasicAuth:

On failed authentication Flask-BasicAuth replies with a 401: Unauthorized, though not in Json.