

CPSC 449-01 Project 2: Microblog Service
Frank Mirando, mirandofrank@csu.fullerton.edu
Ivan Parrales, iparrales@csu.fullerton.edu
Mina Moslehpour, mmoslehpour@csu.fullerton.edu

The following table is the USERS table. It is read from left to right. Take the first row for example. It depicts the information for the user A. A is the username, A@A.com is their email and their password is AA.

USERS TABLE

<u>Username</u>	<u>Email</u>	<u>Password (unhashed)</u>
<u>A</u>	<u>A@A.com</u>	<u>AA</u>
<u>G</u>	<u>G@GGNORE.com</u>	<u>GG</u>
<u>X</u>	<u>X@X.com</u>	<u>XX</u>
<u>B</u>	<u>B@B.com</u>	<u>BB</u>
<u>C</u>	<u>C@C.com</u>	<u>CC</u>
<u>D</u>	<u>D@D.com</u>	<u>DD</u>
<u>F</u>	<u>F@F.com</u>	<u>FF</u>

The following table depicts who a certain user is following. The table is read from right to left. Take the first row for example. User A is following user G. The next row shows that user A is also following user X.

FOLLOWERS TABLE

<u>Following</u>	<u>User</u>
<u>G</u>	<u>A</u>
<u>X</u>	<u>A</u>
<u>E</u>	<u>A</u>
<u>C</u>	<u>B</u>
<u>F</u>	<u>B</u>
<u>A</u>	<u>D</u>
<u>X</u>	<u>G</u>
<u>G</u>	<u>X</u>
<u>A</u>	<u>X</u>

The following table is the Tweets table and shows the text of a tweet, the timestamp of when it was posted and the Author/User who posted the tweet.

TWEETS TABLE

<u>Tweet</u>	<u>Time_Stamp</u>	<u>Author</u>
<u>‘Andromeda’</u>	<u>2020-10-05 19:05:13</u>	<u>A</u>
<u>‘GG NO RE’</u>	<u>2020-10-05 20:05:13</u>	<u>G</u>
<u>‘XANDER CAGE’</u>	<u>2020-10-05 21:05:13</u>	<u>X</u>
<u>‘JUST BO-LIEVE’</u>	<u>2020-10-05 22:05:13</u>	<u>B</u>
<u>‘C WHAT I DID THERE’</u>	<u>2020-10-05 07:05:13</u>	<u>C</u>
<u>‘DINOSAURS ARE DOPE’</u>	<u>2020-10-05 06:05:13</u>	<u>D</u>
<u>‘FANTASTIC’</u>	<u>2020-10-05 05:05:13</u>	<u>F</u>
<u>‘APPLES AND PIES’</u>	<u>2020-10-06 03:12:34</u>	<u>A</u>
<u>‘GEE GEE GEE’</u>	<u>2020-10-06 02:12:34</u>	<u>G</u>
<u>‘X-PAC’</u>	<u>2020-10-07 02:21:43</u>	<u>X</u>
<u>‘KENNY V OKADA II 6.5/5’</u>	<u>2020-10-07 04:43:43</u>	<u>F</u>
<u>‘KENNY V OKADA IV 7/5’</u>	<u>2020-10-07 12:41:33</u>	<u>C</u>

What the Microservices are

User API:

The User API deals mostly with the operations associated with a standard user. The following commands are exposed in the API.

- `createUser(username, email, password)`

Registers a new user account.

- `authenticateUser(username, password)`

Returns true if the supplied password matches the hashed password stored for that username in the database.

- `addFollower(username, usernameToFollow)`

Start following a new user.

- `removeFollower(username, usernameToRemove)`

Stop following a user.

The commands each take in some parameters that will be used when sending requests. The API uses the following HTTP commands when sending the requests. HTTP GET/POST/PUT. By using the commands along with the parameters passed in the functions the API sends a request and receives a response accordingly.

Timeline API:

The Timeline API deals mostly with the operations associated with the tweets and the timeline aka list of specific tweets. The following commands are exposed in the API.

- `getUserTimeline(username)`

Returns recent tweets from a user.

- `getPublicTimeline()`

Returns recent tweets from all users.

- `getHomeTimeline(username)`

Returns recent tweets from all users that this user follows.

- `postTweet(username, text)`

Post a new tweet.

The commands each take in some parameters that will be used when sending requests. The API uses the following HTTP commands when sending the requests. HTTP GET/POST/PUT. By using the commands along with the parameters passed in the functions the API sends a request and receives a response accordingly.

Design

The following will illustrate the design concepts for each of the exposed functions. Each design will show the appropriate JSON data format, HTTP method, URL endpoint, and HTTP status code

User API commands:

- createUser(username, email, password)
 - JSON {
 message = '{username} was added successfully'
}
 - HTTP method: POST
 - URL endpoint: '/register'
 - HTTP status code: 201

- authenticateUser(username, password)
 - JSON {
 message = '{username} was authenticated successfully' (if successful)
 message = '{username} password was incorrect' (if unsuccessful)
}
 - HTTP method: POST
 - URL endpoint: '/loginr'
 - HTTP status code: 201 if successful, 401 is unsuccessful

- `addFollower(username, usernameToFollow)`
 - JSON {
 message = '{username} is now following {usernameToFollow}'
}
 - HTTP method: PUT
 - URL endpoint: '/follow'
 - HTTP status code: 200
- `removeFollower(username, usernameToRemove)`
 - JSON {
 message = '{username} is now unfollowing {usernameToRemove}'
}
 - HTTP method: POST
 - URL endpoint: '/unfollow'
 - HTTP status code: 200

Timeline API Commands

- `getUserTimeline(username)`
 - JSON {
 - [
 - 'Tweet',
 - 'Time stamp',
 - 'Author'
 - (rest of tweets following same format)
 - }
 - HTTP method: GET
 - URL endpoint: `/userTimeline`
 - HTTP status code: 201
- `getPublicTimeline()`
 - JSON {
 - [
 - 'Tweet',
 - 'Time stamp',
 - 'Author'
 - (rest of tweets following same format)
 - }
 - HTTP method: GET
 - URL endpoint: `/publicTimeline`
 - HTTP status code: 201

- `getHomeTimeline(username)`
 - JSON {
 - [
 - 'Tweet',
 - 'Time stamp',
 - 'Author'
 -]
 - (rest of tweets following same format)
 - }
 - HTTP method: GET
 - URL endpoint: '/homeTimeline'
 - HTTP status code: 201

- `postTweet(username, text)`
 - JSON {
 - message = '{username} {tweetText} posted '
 - }
 - HTTP method: POST
 - URL endpoint: '/postTweet'
 - HTTP status code: 200

How To Run Our MicroServices On Tuffix

1. Download project contents into a directory and cd to that directory in your terminal. The project directory should contain the following files:

api_timeline.py

api_users.py

Procfile

README.md

schema.sql

Project 2 Documentation.pdf

2. Run the following export commands in the terminal:

export FLASK_APP=api_timeline.py

export FLASK_APP=api_users.py

3. Run the command (this will create the database): **flask init**

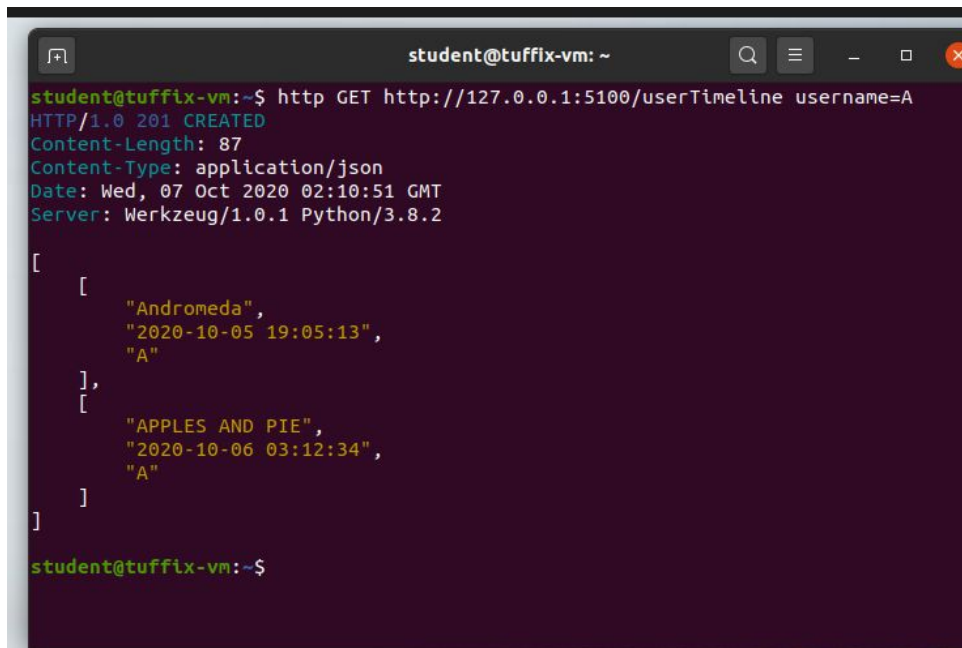
4. Run this command to start the two microservices: **foreman start**

```
api_timeline.py  api_users.py  Procfile  README.md  schema.sql
student@tuffix-vm:~/Desktop/CPSC449(BackEnd)/449-Project-2$ export FLASK_APP=ait_timeline.py
student@tuffix-vm:~/Desktop/CPSC449(BackEnd)/449-Project-2$ export FLASK_APP=api_timeline.py
student@tuffix-vm:~/Desktop/CPSC449(BackEnd)/449-Project-2$ export FLASK_APP=api_users.py
student@tuffix-vm:~/Desktop/CPSC449(BackEnd)/449-Project-2$ flask init
student@tuffix-vm:~/Desktop/CPSC449(BackEnd)/449-Project-2$ foreman start
```

5. Open a new terminal window to execute HTTP commands. The following commands are example test cases and were executed using HTTPie. Each command is followed by a screenshot of the expected output.

- `getUserTimeline(username)`: will get tweets of specified user

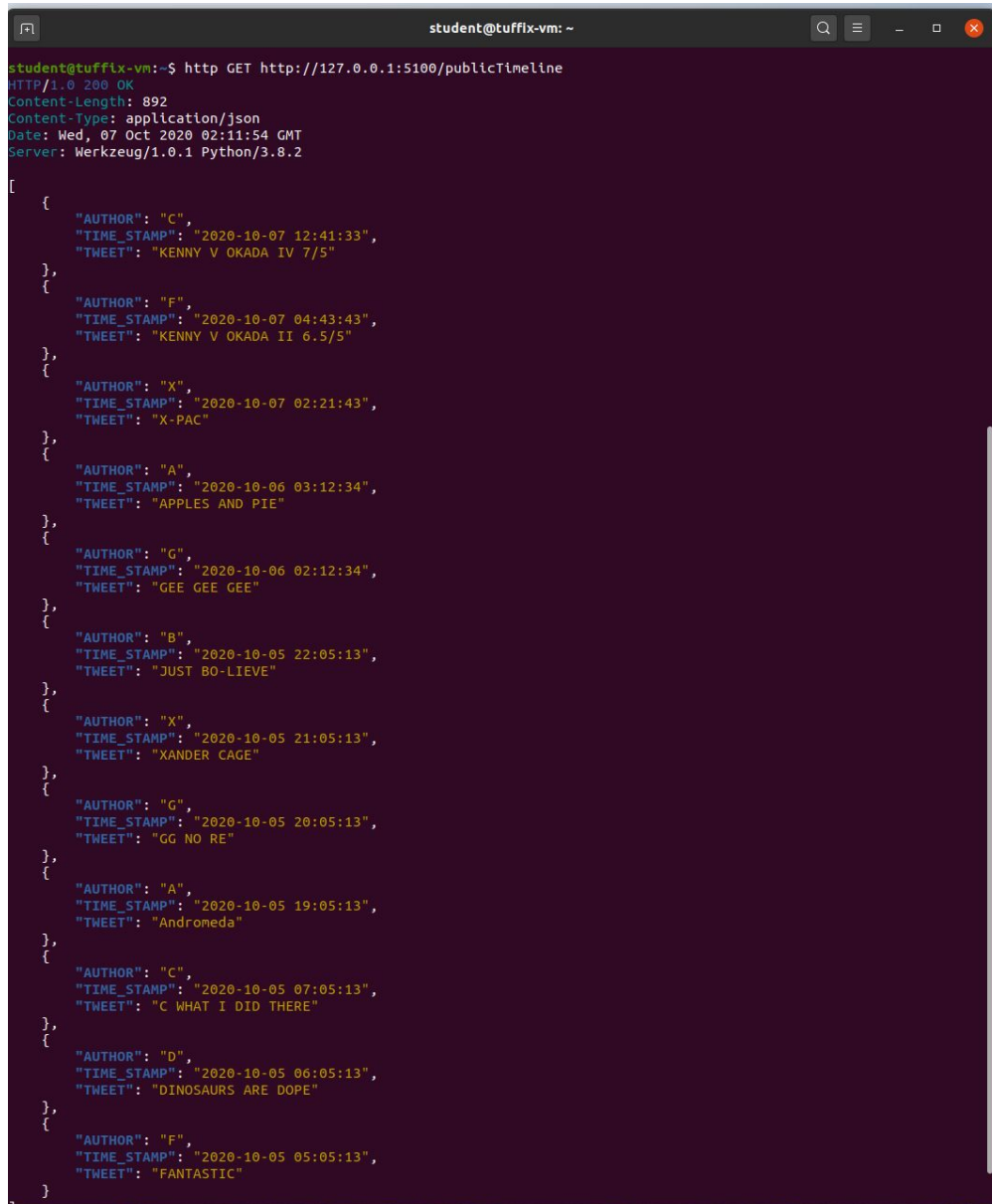
http GET http://127.0.0.1:5100/userTimeline username=A



```
student@tuffix-vm: ~  
student@tuffix-vm:~$ http GET http://127.0.0.1:5100/userTimeline username=A  
HTTP/1.0 201 CREATED  
Content-Length: 87  
Content-Type: application/json  
Date: Wed, 07 Oct 2020 02:10:51 GMT  
Server: Werkzeug/1.0.1 Python/3.8.2  
  
[  
  [  
    "Andromeda",  
    "2020-10-05 19:05:13",  
    "A"  
  ],  
  [  
    "APPLES AND PIE",  
    "2020-10-06 03:12:34",  
    "A"  
  ]  
]  
  
student@tuffix-vm:~$
```

- `getPublicTimeline()`: Displays recent tweets from all users

http GET `http://127.0.0.1:5100/publicTimeline`



```
student@tuffix-vm: ~  
student@tuffix-vm:~$ http GET http://127.0.0.1:5100/publicTimeline  
HTTP/1.0 200 OK  
Content-Length: 892  
Content-Type: application/json  
Date: Wed, 07 Oct 2020 02:11:54 GMT  
Server: Werkzeug/1.0.1 Python/3.8.2  
[  
  {  
    "AUTHOR": "C",  
    "TIME_STAMP": "2020-10-07 12:41:33",  
    "TWEET": "KENNY V OKADA IV 7/5"  
  },  
  {  
    "AUTHOR": "F",  
    "TIME_STAMP": "2020-10-07 04:43:43",  
    "TWEET": "KENNY V OKADA II 6.5/5"  
  },  
  {  
    "AUTHOR": "X",  
    "TIME_STAMP": "2020-10-07 02:21:43",  
    "TWEET": "X-PAC"  
  },  
  {  
    "AUTHOR": "A",  
    "TIME_STAMP": "2020-10-06 03:12:34",  
    "TWEET": "APPLES AND PIE"  
  },  
  {  
    "AUTHOR": "G",  
    "TIME_STAMP": "2020-10-06 02:12:34",  
    "TWEET": "GEE GEE GEE"  
  },  
  {  
    "AUTHOR": "B",  
    "TIME_STAMP": "2020-10-05 22:05:13",  
    "TWEET": "JUST BO-LIEVE"  
  },  
  {  
    "AUTHOR": "X",  
    "TIME_STAMP": "2020-10-05 21:05:13",  
    "TWEET": "XANDER CAGE"  
  },  
  {  
    "AUTHOR": "G",  
    "TIME_STAMP": "2020-10-05 20:05:13",  
    "TWEET": "GG NO RE"  
  },  
  {  
    "AUTHOR": "A",  
    "TIME_STAMP": "2020-10-05 19:05:13",  
    "TWEET": "Andromeda"  
  },  
  {  
    "AUTHOR": "C",  
    "TIME_STAMP": "2020-10-05 07:05:13",  
    "TWEET": "C WHAT I DID THERE"  
  },  
  {  
    "AUTHOR": "D",  
    "TIME_STAMP": "2020-10-05 06:05:13",  
    "TWEET": "DINOSAURS ARE DOPE"  
  },  
  {  
    "AUTHOR": "F",  
    "TIME_STAMP": "2020-10-05 05:05:13",  
    "TWEET": "FANTASTIC"  
  }  
]
```

- `getHomeTimeline(username)`: Gets recent tweets from all users that this user follows

http GET http://127.0.0.1:5100/homeTimeline username=A

```
student@tuffix-vm: ~  
student@tuffix-vm:~$ http GET http://127.0.0.1:5100/homeTimeline username=A  
HTTP/1.0 200 OK  
Content-Length: 434  
Content-Type: application/json  
Date: Wed, 07 Oct 2020 02:12:32 GMT  
Server: Werkzeug/1.0.1 Python/3.8.2  
  
[  
  {  
    "AUTHOR": "F",  
    "TIME_STAMP": "2020-10-07 04:43:43",  
    "TWEET": "KENNY V OKADA II 6.5/5"  
  },  
  {  
    "AUTHOR": "X",  
    "TIME_STAMP": "2020-10-07 02:21:43",  
    "TWEET": "X-PAC"  
  },  
  {  
    "AUTHOR": "G",  
    "TIME_STAMP": "2020-10-06 02:12:34",  
    "TWEET": "GEE GEE GEE"  
  },  
  {  
    "AUTHOR": "X",  
    "TIME_STAMP": "2020-10-05 21:05:13",  
    "TWEET": "XANDER CAGE"  
  },  
  {  
    "AUTHOR": "G",  
    "TIME_STAMP": "2020-10-05 20:05:13",  
    "TWEET": "GG NO RE"  
  },  
  {  
    "AUTHOR": "F",  
    "TIME_STAMP": "2020-10-05 05:05:13",  
    "TWEET": "FANTASTIC"  
  }  
]  
student@tuffix-vm:~$
```

- `postTweet(username, text)`: Posts new tweet

http POST http://127.0.0.1:5100/postTweet username=A tweet=TEST

```
student@tuffix-vm:~$ http POST http://127.0.0.1:5100/postTweet username=0 tweet=TEST
HTTP/1.0 201 CREATED
Content-Length: 29
Content-Type: application/json
Date: Wed, 07 Oct 2020 23:30:05 GMT
Server: Werkzeug/1.0.1 Python/3.8.5

{
  "message": "0 posted: TEST"
}
```

- `createUser(username, email, password)`: Registers new user account

http POST http://127.0.0.1:5000/register username=O email=O@O.COM password=omg

```
student@tuffix-vm:~$ http POST http://127.0.0.1:5000/register username=0 email=0@0.COM password=00
HTTP/1.0 201 CREATED
Content-Length: 45
Content-Type: application/json
Date: Wed, 07 Oct 2020 23:29:02 GMT
Server: Werkzeug/1.0.1 Python/3.8.5

{
  "message": "0 was registered successfully."
}
```

- `authenticateUser(username, password)`: Returns true if the password that was entered matches the password in the database for that user

http POST http://127.0.0.1:5000/login username=O password=omg

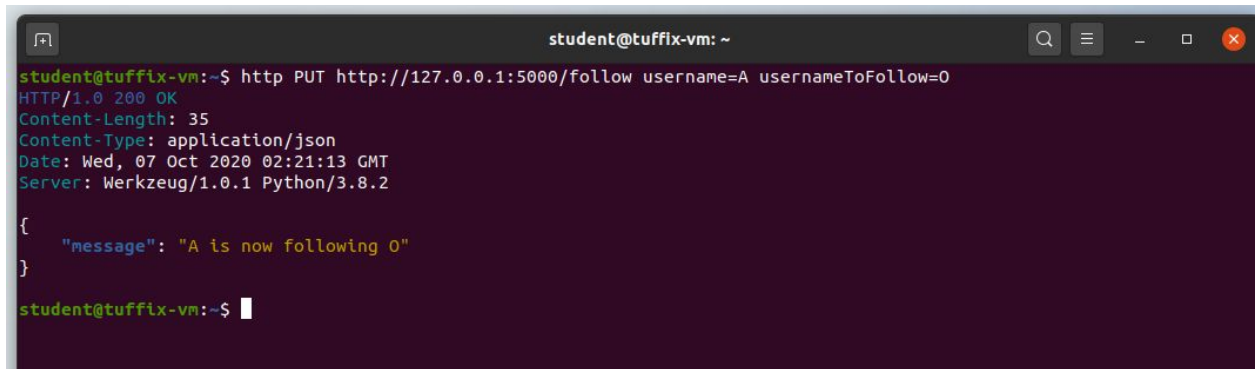
```
student@tuffix-vm:~$ http POST http://127.0.0.1:5000/login username=0 password=omg
HTTP/1.0 201 CREATED
Content-Length: 48
Content-Type: application/json
Date: Wed, 07 Oct 2020 02:19:47 GMT
Server: Werkzeug/1.0.1 Python/3.8.2

{
  "message": "0 was authenticated successfully."
}

student@tuffix-vm:~$
```

- addFollower(username, usernameToFollow): start following a new user

http PUT http://127.0.0.1:5000/follow username=A usernameToFollow=O

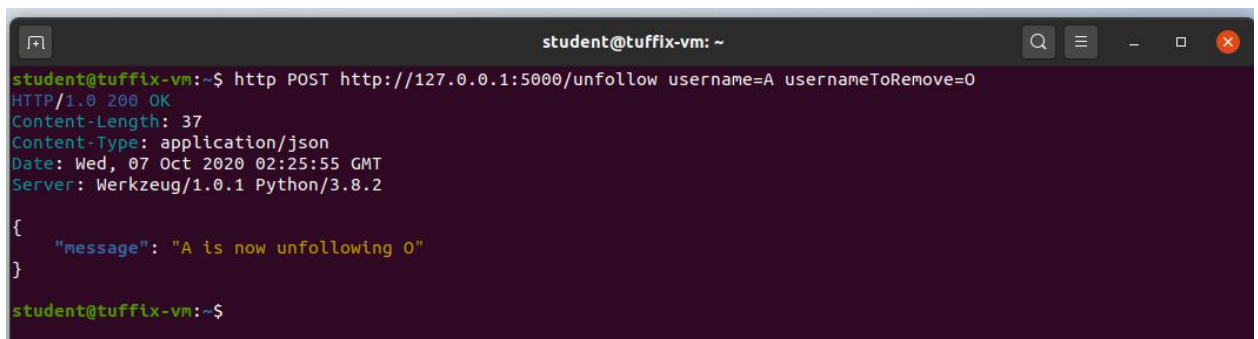
A terminal window titled 'student@tuffix-vm: ~' showing the execution of an HTTP PUT request. The command is 'http PUT http://127.0.0.1:5000/follow username=A usernameToFollow=O'. The output shows a 200 OK status, content length of 35, and a JSON response: {'message': 'A is now following O'}.

```
student@tuffix-vm:~$ http PUT http://127.0.0.1:5000/follow username=A usernameToFollow=O
HTTP/1.0 200 OK
Content-Length: 35
Content-Type: application/json
Date: Wed, 07 Oct 2020 02:21:13 GMT
Server: Werkzeug/1.0.1 Python/3.8.2

{
  "message": "A is now following O"
}
student@tuffix-vm:~$
```

- removeFollower(username, usernameToRemove): Stop following a user

http POST http://127.0.0.1:5000/unfollow username=A usernameToRemove=O

A terminal window titled 'student@tuffix-vm: ~' showing the execution of an HTTP POST request. The command is 'http POST http://127.0.0.1:5000/unfollow username=A usernameToRemove=O'. The output shows a 200 OK status, content length of 37, and a JSON response: {'message': 'A is now unfollowing O'}.

```
student@tuffix-vm:~$ http POST http://127.0.0.1:5000/unfollow username=A usernameToRemove=O
HTTP/1.0 200 OK
Content-Length: 37
Content-Type: application/json
Date: Wed, 07 Oct 2020 02:25:55 GMT
Server: Werkzeug/1.0.1 Python/3.8.2

{
  "message": "A is now unfollowing O"
}
student@tuffix-vm:~$
```