

Testing

To get the two services running use the command

```
$ foreman start
```

Screen should look the same as below

```
student@tuffix-vm:~/cpsc-449$ foreman start
15:53:45 users.1 | started with pid 31905
15:53:45 timeline.1 | started with pid 31906
15:53:49 timeline.1 | * Serving Flask app "timeline_api.py"
15:53:49 timeline.1 | * Environment: production
15:53:49 timeline.1 | WARNING: This is a development server. Do not use it in a production deployment.
15:53:49 timeline.1 | Use a production WSGI server instead.
15:53:49 timeline.1 | * Debug mode: off
15:53:49 users.1 | * Serving Flask app "user_api.py"
15:53:49 users.1 | * Environment: production
15:53:49 users.1 | WARNING: This is a development server. Do not use it in a production deployment.
15:53:49 users.1 | Use a production WSGI server instead.
15:53:49 users.1 | * Debug mode: off
15:53:49 timeline.1 | * Running on http://127.0.0.1:5100/ (Press CTRL+C to quit)
15:53:49 users.1 | * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

To start testing the services first open a new terminal

Testing User Services

- View all the currently registered users with the command

```
$ http GET http://127.0.0.1:5000/users/all
```

output should look like the image below

```
student@tuffix-vm:~$ http GET http://127.0.0.1:5000/users/all
HTTP/1.0 200 OK
Content-Length: 878
Content-Type: application/json
Date: Tue, 06 Oct 2020 23:26:58 GMT
Server: Werkzeug/0.16.1 Python/3.8.2

[
  {
    "EMAIL": "a@a.com",
    "PASSWORD": "pbkdf2:sha256:150000$FAKnujDHSb8d780c1fcbc4eaa5e9d9f9bc9e1046100bb22e04cc58d05c0b9a53746ec4942",
    "PK_USERNAME": "a"
  },
  {
    "EMAIL": "b@b.com",
    "PASSWORD": "pbkdf2:sha256:150000$CZP4YD51$e8016b6bfac95266944f500946392810b1669d6cd4bc5d5f13e0f5bf7c779d2",
    "PK_USERNAME": "b"
  },
  {
    "EMAIL": "c@c.com",
    "PASSWORD": "pbkdf2:sha256:150000$dodjKvIe$950968c0302575bf1936f9a49a14fb194a224e41d9b7a6a426ac22ef9ac85de",
    "PK_USERNAME": "c"
  },
  {
    "EMAIL": "d@d.com",
    "PASSWORD": "pbkdf2:sha256:150000$7H0LmQlJ$351c9b0a097bf1ed0763e6e94c032bb65914b0b6c3604e4130323698bd360184",
    "PK_USERNAME": "d"
  },
  {
    "EMAIL": "e@e.com",
    "PASSWORD": "pbkdf2:sha256:150000$LpuqQylz$6448a105414f1e53859
student@tuffix-vm:~/cpsc-449$ foreman start
16:25:01 users.1 | started with pid 32035
16:25:01 timeline.1 | started with pid 32036
16:25:05 timeline.1 | * Serving Flask app "timeline_api.py"
16:25:05 timeline.1 | * Environment: production
16:25:05 timeline.1 | WARNING: This is a development server. Do not use it in a production deployment.
16:25:05 timeline.1 | Use a production WSGI server instead.
16:25:05 timeline.1 | * Debug mode: off
16:25:05 users.1 | * Serving Flask app "user_api.py"
16:25:05 users.1 | * Environment: production
16:25:05 users.1 | WARNING: This is a development server. Do not use it in a production deployment.
16:25:05 users.1 | Use a production WSGI server instead.
16:25:05 users.1 | * Debug mode: off
16:25:05 timeline.1 | * Running on http://127.0.0.1:5100/ (Press CTRL+C to quit)
16:25:05 users.1 | * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
16:26:58 users.1 | 127.0.0.1 - - [06/Oct/2020 16:26:58] "GET /users/all HTTP/1.1" 200 -
```

Note: not all users shown on screenshot.

- View the users that a user is currently following you use the command,

\$ http GET http://127.0.0.1:5000/following username=NAME

Where NAME is the username of the person that you want to see who they are following. Output should look like the image bellow where username=a

```
student@tuffix-vm:~$ http GET http://127.0.0.1:5000/following username=a
HTTP/1.0 200 OK
Content-Length: 98
Content-Type: application/json
Date: Tue, 06 Oct 2020 23:41:33 GMT
Server: Werkzeug/0.16.1 Python/3.8.2

[
  {
    "FK_USER": "a",
    "FOLLOWERS": "b"
  },
  {
    "FK_USER": "a",
    "FOLLOWERS": "c"
  },
  {
    "FK_USER": "a",
    "FOLLOWERS": "d"
  }
]

student@tuffix-vm:~/cpsc-449$ foreman start
16:41:13 users.1 | started with pid 32102
16:41:13 timeline.1 | started with pid 32103
16:41:16 users.1 | * Serving Flask app "user_api.py"
16:41:16 users.1 | * Environment: production
16:41:16 users.1 | WARNING: This is a development server. Do not
                  | use it in a production deployment.
16:41:16 users.1 | Use a production WSGI server instead.
16:41:16 users.1 | * Debug mode: off
16:41:16 timeline.1 | * Serving Flask app "timeline_api.py"
16:41:16 timeline.1 | * Environment: production
16:41:16 timeline.1 | WARNING: This is a development server. Do not
                  | use it in a production deployment.
16:41:16 timeline.1 | Use a production WSGI server instead.
16:41:16 timeline.1 | * Debug mode: off
16:41:16 timeline.1 | * Running on http://127.0.0.1:5100/ (Press CTRL
                  | +C to quit)
16:41:16 users.1 | * Running on http://127.0.0.1:5000/ (Press CTRL
                  | +C to quit)
16:41:33 users.1 | 127.0.0.1 - - [06/Oct/2020 16:41:33] "GET /follo
                  | wing HTTP/1.1" 200 -
```

- Create a user you use the following command,

\$ http POST http://127.0.0.1:5000/register username=NAME email=EMAIL
password=PASSWORD

Where NAME is the username of the person, EMAIL is the email, and PASSWORD is their password which will be hashed. Output should look like the image below where username=z, email=z@z.com, password=z

```
student@tuffix-vm:~$ http POST http://127.0.0.1:5000/register username=z email=z@z.com password=z
HTTP/1.0 201 CREATED
Content-Length: 40
Content-Type: application/json
Date: Tue, 06 Oct 2020 23:32:49 GMT
Server: Werkzeug/0.16.1 Python/3.8.2

{
  "message": "z was added successfully."
}

student@tuffix-vm:~$

student@tuffix-vm:~/cpsc-449$ foreman start
16:25:01 users.1 | started with pid 32035
16:25:01 timeline.1 | started with pid 32036
16:25:05 timeline.1 | * Serving Flask app "timeline_api.py"
16:25:05 timeline.1 | * Environment: production
16:25:05 timeline.1 | WARNING: This is a development server. Do not
                  | use it in a production deployment.
16:25:05 timeline.1 | Use a production WSGI server instead.
16:25:05 timeline.1 | * Debug mode: off
16:25:05 users.1 | * Serving Flask app "user_api.py"
16:25:05 users.1 | * Environment: production
16:25:05 users.1 | WARNING: This is a development server. Do not
                  | use it in a production deployment.
16:25:05 users.1 | Use a production WSGI server instead.
16:25:05 users.1 | * Debug mode: off
16:25:05 timeline.1 | * Running on http://127.0.0.1:5100/ (Press CTRL
                  | +C to quit)
16:25:05 users.1 | * Running on http://127.0.0.1:5000/ (Press CTRL
                  | +C to quit)
16:26:58 users.1 | 127.0.0.1 - - [06/Oct/2020 16:26:58] "GET /users
                  | /all HTTP/1.1" 200 -
16:32:49 users.1 | 127.0.0.1 - - [06/Oct/2020 16:32:49] "POST /regl
                  | ster HTTP/1.1" 201 -
```

Note: to verify that a user has been updated to the database you can run the /users/all cmd and scroll to the bottom where the new user should be located

- Authenticate a user you use the following command,

\$ http POST http://127.0.0.1:5000/login username=NAME password=PASSWORD

Where NAME is the username of the person to be authenticated and PASSWORD is the persons password. Output should look like the image bellow where username=a and password=a

```
student@tuffix-vm:~$ http POST http://127.0.0.1:5000/login username=a password=a
HTTP/1.0 201 CREATED
Content-Length: 48
Content-Type: application/json
Date: Tue, 06 Oct 2020 23:52:55 GMT
Server: Werkzeug/0.16.1 Python/3.8.2

{
  "message": "a was authenticated successfully."
}
student@tuffix-vm:~$
```

```
student@tuffix-vm:~/cpsc-449$ foreman start
16:50:08 users.1 | started with pid 32196
16:50:08 timeline.1 | started with pid 32197
16:50:12 users.1 | * Serving Flask app "user_api.py"
16:50:12 users.1 | * Environment: production
16:50:12 users.1 | WARNING: This is a development server. Do not
                  | use it in a production deployment.
16:50:12 users.1 | Use a production WSGI server instead.
16:50:12 users.1 | Debug mode: off
16:50:12 timeline.1 | * Serving Flask app "timeline_api.py"
16:50:12 timeline.1 | * Environment: production
16:50:12 timeline.1 | WARNING: This is a development server. Do not
                  | use it in a production deployment.
16:50:12 timeline.1 | Use a production WSGI server instead.
16:50:12 timeline.1 | Debug mode: off
16:50:12 users.1 | * Running on http://127.0.0.1:5000/ (Press CTRL
                  | +C to quit)
16:50:12 timeline.1 | * Running on http://127.0.0.1:5100/ (Press CTRL
                  | +C to quit)
16:51:52 users.1 | 127.0.0.1 - - [06/Oct/2020 16:51:52] "GET /users
                  | /all HTTP/1.1" 200 -
16:52:04 users.1 | 127.0.0.1 - - [06/Oct/2020 16:52:04] "GET /follo
                  | wing HTTP/1.1" 200 -
16:52:14 users.1 | 127.0.0.1 - - [06/Oct/2020 16:52:14] "POST /regi
                  | ster HTTP/1.1" 201 -
16:52:21 users.1 | 127.0.0.1 - - [06/Oct/2020 16:52:21] "GET /users
                  | /all HTTP/1.1" 200 -
16:52:55 users.1 | 127.0.0.1 - - [06/Oct/2020 16:52:55] "POST /logi
                  | n HTTP/1.1" 201 -
```

If a user tries to authenticate with the wrong password it will say incorrect password as can be seen below where username=a and password=b

```
student@tuffix-vm:~$ http POST http://127.0.0.1:5000/login username=a password=b
HTTP/1.0 401 UNAUTHORIZED
Content-Length: 35
Content-Type: application/json
Date: Tue, 06 Oct 2020 23:56:10 GMT
Server: Werkzeug/0.16.1 Python/3.8.2

{
  "message": "a password incorrect"
}
student@tuffix-vm:~$
```

```
student@tuffix-vm:~/cpsc-449$ foreman start
16:50:08 users.1 | started with pid 32196
16:50:08 timeline.1 | started with pid 32197
16:50:12 users.1 | * Serving Flask app "user_api.py"
16:50:12 users.1 | * Environment: production
16:50:12 users.1 | WARNING: This is a development server. Do not
                  | use it in a production deployment.
16:50:12 users.1 | Use a production WSGI server instead.
16:50:12 users.1 | Debug mode: off
16:50:12 timeline.1 | * Serving Flask app "timeline_api.py"
16:50:12 timeline.1 | * Environment: production
16:50:12 timeline.1 | WARNING: This is a development server. Do not
                  | use it in a production deployment.
16:50:12 timeline.1 | Use a production WSGI server instead.
16:50:12 timeline.1 | Debug mode: off
16:50:12 users.1 | * Running on http://127.0.0.1:5000/ (Press CTRL
                  | +C to quit)
16:50:12 timeline.1 | * Running on http://127.0.0.1:5100/ (Press CTRL
                  | +C to quit)
16:51:52 users.1 | 127.0.0.1 - - [06/Oct/2020 16:51:52] "GET /users
                  | /all HTTP/1.1" 200 -
16:52:04 users.1 | 127.0.0.1 - - [06/Oct/2020 16:52:04] "GET /follo
                  | wing HTTP/1.1" 200 -
16:52:14 users.1 | 127.0.0.1 - - [06/Oct/2020 16:52:14] "POST /regi
                  | ster HTTP/1.1" 201 -
16:52:21 users.1 | 127.0.0.1 - - [06/Oct/2020 16:52:21] "GET /users
                  | /all HTTP/1.1" 200 -
16:52:55 users.1 | 127.0.0.1 - - [06/Oct/2020 16:52:55] "POST /logi
                  | n HTTP/1.1" 201 -
16:56:10 users.1 | 127.0.0.1 - - [06/Oct/2020 16:56:10] "POST /logi
                  | n HTTP/1.1" 401 -
```

- Add a follower you use the following command,

\$ http PUT http://127.0.0.1:5000/follow username=NAME usernameToFollow=USER

Where NAME is the username of the person who wants to follow USER. Output should look like the image bellow where username=a and usernameToFollow=z

```
student@tuffix-vm:~$ http PUT http://127.0.0.1:5000/follow username=a
usernameToFollow=z
HTTP/1.0 200 OK
Content-Length: 35
Content-Type: application/json
Date: Wed, 07 Oct 2020 00:04:41 GMT
Server: Werkzeug/0.16.1 Python/3.8.2

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

```
student@tuffix-vm:~/cpsc-449$ foreman start
17:03:53 users.1 | started with pid 32328
17:03:53 timeline.1 | started with pid 32329
17:03:56 users.1 | * Serving Flask app "user_api.py"
17:03:56 users.1 | * Environment: production
17:03:56 users.1 | WARNING: This is a development server. Do not
17:03:56 users.1 | use it in a production deployment.
17:03:56 users.1 | Use a production WSGI server instead.
17:03:56 users.1 | * Debug mode: off
17:03:56 timeline.1 | * Serving Flask app "timeline_api.py"
17:03:56 timeline.1 | * Environment: production
17:03:56 timeline.1 | WARNING: This is a development server. Do not
17:03:56 timeline.1 | use it in a production deployment.
17:03:56 timeline.1 | Use a production WSGI server instead.
17:03:56 timeline.1 | * Debug mode: off
17:03:56 users.1 | * Running on http://127.0.0.1:5000/ (Press CTRL
17:03:56 users.1 | +C to quit)
17:03:56 timeline.1 | * Running on http://127.0.0.1:5100/ (Press CTRL
17:03:56 timeline.1 | +C to quit)
17:04:01 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:01] "GET /users
17:04:10 users.1 | /all HTTP/1.1" 200 -
17:04:10 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:10] "POST /regi
17:04:20 users.1 | ster HTTP/1.1" 201 -
17:04:20 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:20] "POST /logi
17:04:28 users.1 | n HTTP/1.1" 201 -
17:04:28 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:28] "POST /logi
17:04:41 users.1 | n HTTP/1.1" 401 -
17:04:41 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:41] "PUT /follo
17:04:41 users.1 | w HTTP/1.1" 200 -
```

Note: to see that a is now actually following z you can use the /following command with username=a to view that z is now being followed by a

- Remove a follower you use the command,

\$ http POST http://127.0.0.1:5000/unfollow username=NAME
usernameToRemove=USER

Where NAME is the username of the person who wants to unfollow USER. Output should look like the image bellow where username=a and usernameToRemove=z

```
student@tuffix-vm:~$ http POST http://127.0.0.1:5000/unfollow username=a
usernameToRemove=z
HTTP/1.0 200 OK
Content-Length: 37
Content-Type: application/json
Date: Wed, 07 Oct 2020 00:09:47 GMT
Server: Werkzeug/0.16.1 Python/3.8.2

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

```
17:03:56 users.1 | * Serving Flask app "user_api.py"
17:03:56 users.1 | * Environment: production
17:03:56 users.1 | WARNING: This is a development server. Do not
17:03:56 users.1 | use it in a production deployment.
17:03:56 users.1 | Use a production WSGI server instead.
17:03:56 users.1 | * Debug mode: off
17:03:56 timeline.1 | * Serving Flask app "timeline_api.py"
17:03:56 timeline.1 | * Environment: production
17:03:56 timeline.1 | WARNING: This is a development server. Do not
17:03:56 timeline.1 | use it in a production deployment.
17:03:56 timeline.1 | Use a production WSGI server instead.
17:03:56 timeline.1 | * Debug mode: off
17:03:56 users.1 | * Running on http://127.0.0.1:5000/ (Press CTRL
17:03:56 users.1 | +C to quit)
17:03:56 timeline.1 | * Running on http://127.0.0.1:5100/ (Press CTRL
17:03:56 timeline.1 | +C to quit)
17:04:01 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:01] "GET /users
17:04:10 users.1 | /all HTTP/1.1" 200 -
17:04:10 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:10] "POST /regi
17:04:20 users.1 | ster HTTP/1.1" 201 -
17:04:20 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:20] "POST /logi
17:04:28 users.1 | n HTTP/1.1" 201 -
17:04:28 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:28] "POST /logi
17:04:41 users.1 | n HTTP/1.1" 401 -
17:04:41 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:41] "PUT /follo
17:04:41 users.1 | w HTTP/1.1" 200 -
17:06:21 users.1 | 127.0.0.1 - - [06/Oct/2020 17:06:21] "GET /follo
17:06:21 users.1 | wing HTTP/1.1" 200 -
17:07:58 users.1 | 127.0.0.1 - - [06/Oct/2020 17:07:58] "POST /foll
17:07:58 users.1 | ow HTTP/1.1" 405 -
17:08:14 users.1 | 127.0.0.1 - - [06/Oct/2020 17:08:14] "POST /unfo
17:08:14 users.1 | llow HTTP/1.1" 200 -
17:09:47 users.1 | 127.0.0.1 - - [06/Oct/2020 17:09:47] "POST /unfo
17:09:47 users.1 | llow HTTP/1.1" 200 -
```

Note: to see that a is now actually unfollowing z you can use the /following command with username=a to view that z is now no longer being followed by a

Testing Timeline Services

- View a user timeline you use the command,

\$ http GET http://127.0.0.1:5100/userTimeline username=Name

Where NAME is the username of the person whose tweets you want to view. Output should look like the image below where username=a

```
student@tuffix-vm:~$ http GET http://127.0.0.1:5100/userTimeline username=a
HTTP/1.0 201 CREATED
Content-Length: 86
Content-Type: application/json
Date: Wed, 07 Oct 2020 00:14:05 GMT
Server: Werkzeug/0.16.1 Python/3.8.2

[
  [
    "2hello all",
    "2020-10-05 10:37:00",
    "a"
  ],
  [
    "8goodbye all",
    "2020-05-05 23:59:59",
    "a"
  ]
]

17:03:56 users.1 | Use a production WSGI server instead.
17:03:56 users.1 | * Debug mode: off
17:03:56 timeline.1 | * Serving Flask app "timeline_api.py"
17:03:56 timeline.1 | * Environment: production
17:03:56 timeline.1 | WARNING: This is a development server. Do not
17:03:56 timeline.1 | use it in a production deployment.
17:03:56 timeline.1 | Use a production WSGI server instead.
17:03:56 timeline.1 | * Debug mode: off
17:03:56 users.1 | * Running on http://127.0.0.1:5000/ (Press CTRL
17:03:56 timeline.1 | +C to quit)
17:03:56 timeline.1 | * Running on http://127.0.0.1:5100/ (Press CTRL
17:03:56 timeline.1 | +C to quit)
17:04:01 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:01] "GET /users
17:04:10 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:10] "POST /regl
17:04:20 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:20] "POST /logi
17:04:28 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:28] "POST /logi
17:04:41 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:41] "PUT /follo
17:06:21 users.1 | 127.0.0.1 - - [06/Oct/2020 17:06:21] "GET /follo
17:07:58 users.1 | 127.0.0.1 - - [06/Oct/2020 17:07:58] "POST /foll
17:08:14 users.1 | 127.0.0.1 - - [06/Oct/2020 17:08:14] "POST /unfo
17:09:47 users.1 | 127.0.0.1 - - [06/Oct/2020 17:09:47] "POST /unfo
17:11:42 users.1 | 127.0.0.1 - - [06/Oct/2020 17:11:42] "GET /follo
17:14:05 timeline.1 | 127.0.0.1 - - [06/Oct/2020 17:14:05] "GET /userT
17:14:05 timeline.1 | timeline HTTP/1.1" 201 -
```

Note: tweets will be output in reverse chronological order based on the timestamp and only the most recent 25 will show

- View the public timeline which will return the recent tweets from all users, you use the command,

\$ http GET http://127.0.0.1:5100/publicTimeline

Output should look like the image below

```
student@tuffix-vm:~$ http GET http://127.0.0.1:5100/publicTimeline
HTTP/1.0 200 OK
Content-Length: 898
Content-Type: application/json
Date: Wed, 07 Oct 2020 00:20:07 GMT
Server: Werkzeug/0.16.1 Python/3.8.2

[
  {
    "DAY_OF": "2059-10-05 14:12:35",
    "FK_USERS": "e",
    "TWEET": "12my time machine worked"
  },
  {
    "DAY_OF": "2020-10-05 14:12:35",
    "FK_USERS": "d",
    "TWEET": "ihello a"
  },
  {
    "DAY_OF": "2020-10-05 10:37:00",
    "FK_USERS": "a",
    "TWEET": "2hello all"
  },
  {
    "DAY_OF": "2020-10-04 17:12:36",
    "FK_USERS": "e",
    "TWEET": "3making them apt"
  },
  {
    "DAY_OF": "2020-10-04 17:12:35",
    "FK_USERS": "c",
    "TWEET": "4c is cool"
  },
  {
    "DAY_OF": "2020-10-04 17:12:34",
    "FK_USERS": "b",
    "TWEET": "7b is cool"
  },
  {
    "DAY_OF": "2019-10-05 14:12:35",
    "FK_USERS": "c",
    "TWEET": "9b is wack"
  },
  {
    "DAY_OF": "2014-10-05 14:12:35",
    "FK_USERS": "d",
    "TWEET": "10d is best"
  }
]
```

```
17:03:56 timeline.1 | * Serving Flask app "timeline_api.py"
17:03:56 timeline.1 | * Environment: production
17:03:56 timeline.1 | WARNING: This is a development server. Do not
17:03:56 timeline.1 | use it in a production deployment.
17:03:56 timeline.1 | Use a production WSGI server instead.
17:03:56 timeline.1 | * Debug mode: off
17:03:56 users.1 | * Running on http://127.0.0.1:5000/ (Press CTRL
17:03:56 users.1 | +C to quit)
17:03:56 timeline.1 | * Running on http://127.0.0.1:5100/ (Press CTRL
17:03:56 timeline.1 | +C to quit)
17:04:01 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:01] "GET /users
17:04:10 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:10] "POST /regi
17:04:20 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:20] "POST /logi
17:04:28 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:28] "POST /logi
17:04:41 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:41] "PUT /follo
17:06:21 users.1 | 127.0.0.1 - - [06/Oct/2020 17:06:21] "GET /follo
17:07:58 users.1 | 127.0.0.1 - - [06/Oct/2020 17:07:58] "POST /foll
17:08:14 users.1 | 127.0.0.1 - - [06/Oct/2020 17:08:14] "POST /unfo
17:09:47 users.1 | 127.0.0.1 - - [06/Oct/2020 17:09:47] "POST /unfo
17:11:42 users.1 | 127.0.0.1 - - [06/Oct/2020 17:11:42] "GET /follo
17:14:05 timeline.1 | 127.0.0.1 - - [06/Oct/2020 17:14:05] "GET /userT
17:20:07 timeline.1 | 127.0.0.1 - - [06/Oct/2020 17:20:07] "GET /publi
17:23:21 timeline.1 | 127.0.0.1 - - [06/Oct/2020 17:23:21] "GET /homeT
```

Note: tweets will be output in reverse chronological order based on the timestamp and only the most recent 25 will show

- View the home timeline of a person which will output only tweets from the people that the person follows you use the command,

\$ http GET http://127.0.0.1:5100/homeTimeline username=NAME

Where Name is the username of the person that you want to get the home timeline of, output should look like the image below

```
student@tuffix-vm:~$ http GET http://127.0.0.1:5100/homeTimeline username=a
HTTP/1.0 200 OK
Content-Length: 346
Content-Type: application/json
Date: Wed, 07 Oct 2020 00:23:21 GMT
Server: Werkzeug/0.16.1 Python/3.8.2

[
  {
    "DAY_OF": "2020-10-05 14:12:35",
    "FK_USERS": "d",
    "TWEET": "ihello a"
  },
  {
    "DAY_OF": "2020-10-04 17:12:35",
    "FK_USERS": "c",
    "TWEET": "4c is cool"
  },
  {
    "DAY_OF": "2020-09-28 05:58:59",
    "FK_USERS": "b",
    "TWEET": "7b is cool"
  },
  {
    "DAY_OF": "2019-10-05 14:12:35",
    "FK_USERS": "c",
    "TWEET": "9b is wack"
  },
  {
    "DAY_OF": "2014-10-05 14:12:35",
    "FK_USERS": "d",
    "TWEET": "10d is best"
  }
]
```

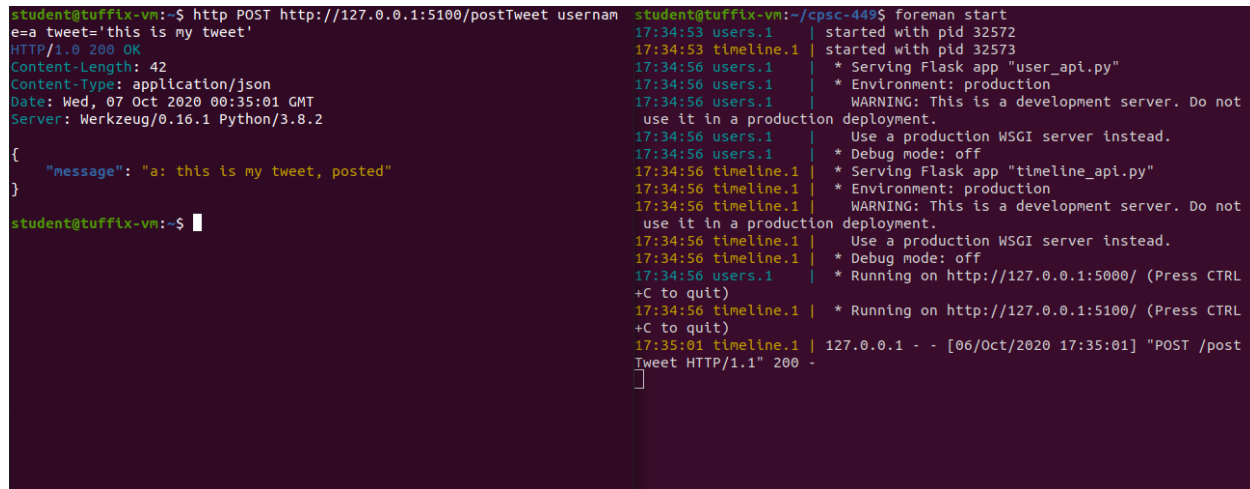
```
17:03:56 timeline.1 | WARNING: This is a development server. Do not
17:03:56 timeline.1 | use it in a production deployment.
17:03:56 timeline.1 | Use a production WSGI server instead.
17:03:56 timeline.1 | * Debug mode: off
17:03:56 users.1 | * Running on http://127.0.0.1:5000/ (Press CTRL
17:03:56 users.1 | +C to quit)
17:03:56 timeline.1 | * Running on http://127.0.0.1:5100/ (Press CTRL
17:03:56 timeline.1 | +C to quit)
17:04:01 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:01] "GET /users
17:04:10 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:10] "POST /regi
17:04:20 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:20] "POST /logi
17:04:28 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:28] "POST /logi
17:04:41 users.1 | 127.0.0.1 - - [06/Oct/2020 17:04:41] "PUT /follo
17:06:21 users.1 | 127.0.0.1 - - [06/Oct/2020 17:06:21] "GET /follo
17:07:58 users.1 | 127.0.0.1 - - [06/Oct/2020 17:07:58] "POST /foll
17:08:14 users.1 | 127.0.0.1 - - [06/Oct/2020 17:08:14] "POST /unfo
17:09:47 users.1 | 127.0.0.1 - - [06/Oct/2020 17:09:47] "POST /unfo
17:11:42 users.1 | 127.0.0.1 - - [06/Oct/2020 17:11:42] "GET /follo
17:14:05 timeline.1 | 127.0.0.1 - - [06/Oct/2020 17:14:05] "GET /userT
17:20:07 timeline.1 | 127.0.0.1 - - [06/Oct/2020 17:20:07] "GET /publi
17:23:21 timeline.1 | 127.0.0.1 - - [06/Oct/2020 17:23:21] "GET /homeT
```


Note: tweets will be output in reverse chronological order based on the timestamp and only the most recent 25 will show

- Post a tweet you use the command,

\$ http POST http://127.0.0.1:5100/postTweet username=NAME tweet='TEXT'

Where NAME is the username of the person posting and 'TEXT' is the text of the tweet within single quotes, output should look like the image below



```
student@tuffix-vm:~$ http POST http://127.0.0.1:5100/postTweet username=a tweet='this is my tweet'
HTTP/1.0 200 OK
Content-Length: 42
Content-Type: application/json
Date: Wed, 07 Oct 2020 00:35:01 GMT
Server: Werkzeug/0.16.1 Python/3.8.2

{
  "message": "a: this is my tweet, posted"
}
student@tuffix-vm:~$

student@tuffix-vm:~/cpsc-449$ foreman start
17:34:53 users.1 | started with pid 32572
17:34:53 timeline.1 | started with pid 32573
17:34:56 users.1 | * Serving Flask app "user_api.py"
17:34:56 users.1 | * Environment: production
17:34:56 users.1 | WARNING: This is a development server. Do not
                  | use it in a production deployment.
                  | Use a production WSGI server instead.
17:34:56 users.1 | * Debug mode: off
17:34:56 timeline.1 | * Serving Flask app "timeline_api.py"
17:34:56 timeline.1 | * Environment: production
17:34:56 timeline.1 | WARNING: This is a development server. Do not
                  | use it in a production deployment.
                  | Use a production WSGI server instead.
17:34:56 timeline.1 | * Debug mode: off
17:34:56 users.1 | * Running on http://127.0.0.1:5000/ (Press CTRL
                  | +C to quit)
17:34:56 timeline.1 | * Running on http://127.0.0.1:5100/ (Press CTRL
                  | +C to quit)
17:35:01 timeline.1 | 127.0.0.1 - - [06/Oct/2020 17:35:01] "POST /post
                  | Tweet HTTP/1.1" 200 -
```

Note: to verify that the tweet was correctly posted and added to the database you could run the /publicTimeline command and the newly added tweet should be the second item on the list since they are added in reverse chronological order and one of the manually inserted tweets will always be the first due to the timestamp it posses