



COLLEGE OF ENGINEERING AND COMPUTER SCIENCE

Fall 2020

CPSC 449 Web-Backend

Project 4 - Documentation

Report Prepared By:

<u>CWID</u>	<u>Name</u>
<u>887480747</u>	<u>Venkata Pranathi Immaneni</u>

Table of Contents

1	<i>How to run the Project?</i>	3
2.	<i>Flow Diagram</i>	4
3.	<i>Introducing API Gateway</i>	6
4.	<i>Changes in Users Microservice</i>	7
2.1	createUser API:	7
2.2	authenticateUser API:	7
2.3	addFollower API:	8
2.4	removeFollower API:	9
5	<i>Changes in Timeline Microservice</i>	11
3.1	postTweet API:	11
3.2	getUserTimeline API:	12
3.3	getPublicTimeline API:	13
3.4	getHomeTimeline API:	14

1 How to run the Project?

Step 1: Run the below command to initialise the database

```
flask init
```

Step 2: Run the below command to start the 3 instances of Users service, 3 instance of Timelines Service and 1 instance of gateway.

```
foreman start -m gateway=1,Usersapi=3,Timelinesapi=3
```

Note:

Gateway runs on 5000 Port

Usersapi runs on 5100, 5101, 5102 Ports

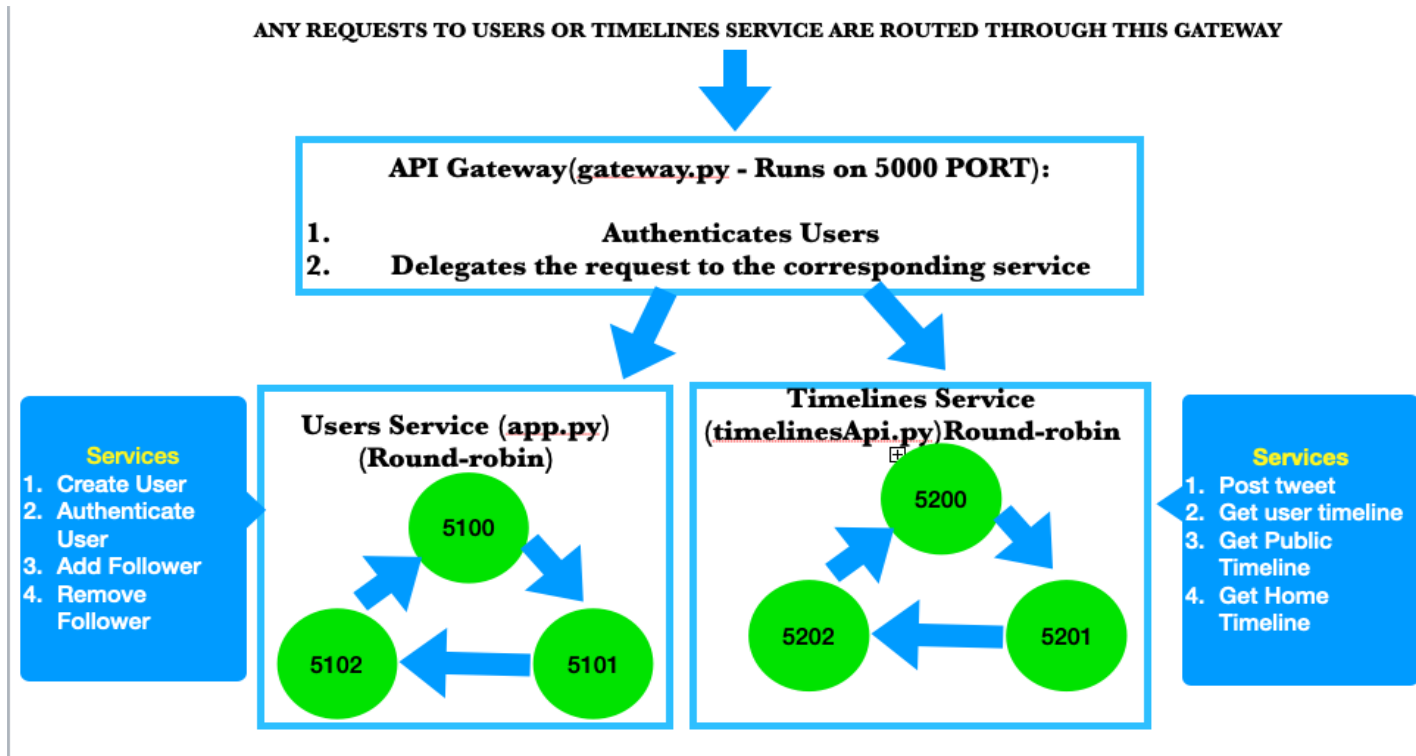
Timelinesapi runs on 5200, 5201, 5202 Ports

Below screenshots depicts the same:

```
student@tufflix-vm:~/Desktop/Project4$ foreman start -m gateway=1,Usersapi=3,Timelinesapi=3
19:16:47 gateway.1      | started with pid 31742
19:16:47 Usersapi.1     | started with pid 31743
19:16:47 Usersapi.2     | started with pid 31745
19:16:47 Usersapi.3     | started with pid 31747
19:16:47 Timelinesapi.1 | started with pid 31749
19:16:47 Timelinesapi.2 | started with pid 31751
19:16:47 Timelinesapi.3 | started with pid 31753
19:16:49 Usersapi.1     | * Serving Flask app "app" (lazy loading)
19:16:49 Timelinesapi.1 | * Serving Flask app "timelinesApi" (lazy loading)
19:16:49 Usersapi.2     | * Serving Flask app "app" (lazy loading)
19:16:49 Usersapi.2     | * Environment: development
19:16:49 Usersapi.2     | * Debug mode: on
19:16:49 Usersapi.3     | * Serving Flask app "app" (lazy loading)
19:16:49 Usersapi.3     | * Environment: development
19:16:49 Usersapi.3     | * Debug mode: on
19:16:49 gateway.1     | * Serving Flask app "gateway" (lazy loading)
19:16:49 gateway.1     | * Environment: development
19:16:49 gateway.1     | * Debug mode: on
19:16:49 Timelinesapi.3 | * Serving Flask app "timelinesApi" (lazy loading)
19:16:49 Timelinesapi.3 | * Environment: development
19:16:49 Timelinesapi.3 | * Debug mode: on
19:16:49 Timelinesapi.1 | * Environment: development
```

```
19:16:49 Timelinesapi.1 | * Running on http://127.0.0.1:5200/ (Press CTRL+C to quit)
19:16:49 Timelinesapi.1 | * Restarting with stat
19:16:49 Usersapi.2 | * Running on http://127.0.0.1:5101/ (Press CTRL+C to quit)
19:16:49 Usersapi.2 | * Restarting with stat
19:16:49 Timelinesapi.3 | * Running on http://127.0.0.1:5202/ (Press CTRL+C to quit)
19:16:49 Timelinesapi.3 | * Restarting with stat
19:16:49 gateway.1 | * Restarting with stat
19:16:49 Usersapi.1 | * Running on http://127.0.0.1:5100/ (Press CTRL+C to quit)
19:16:49 Usersapi.3 | * Running on http://127.0.0.1:5102/ (Press CTRL+C to quit)
19:16:49 Usersapi.3 | * Restarting with stat
19:16:49 Usersapi.1 | * Restarting with stat
19:16:49 Timelinesapi.2 | * Running on http://127.0.0.1:5201/ (Press CTRL+C to quit)
19:16:49 Timelinesapi.2 | * Restarting with stat
19:16:50 Usersapi.1 | * Debugger is active!
19:16:50 Usersapi.1 | * Debugger PIN: 516-355-886
19:16:50 Timelinesapi.1 | * Debugger is active!
19:16:50 Timelinesapi.1 | * Debugger PIN: 516-355-886
19:16:50 Timelinesapi.3 | * Debugger is active!
19:16:50 Timelinesapi.3 | * Debugger PIN: 516-355-886
19:16:50 Usersapi.3 | * Debugger is active!
19:16:50 Usersapi.3 | * Debugger PIN: 516-355-886
19:16:51 gateway.1 | * Debugger is active!
19:16:51 Timelinesapi.2 | * Debugger is active!
```

2. Flow Diagram



3. Introducing API Gateway

ALL USERNAMES AND PASSWORDS ARE CASE SENSITIVE.
--

File Name: gateway.py

- **All the requests to Users Services and Timelines Services are routed through this Gateway.**

Primary Functionalities of Gateway.py:

1. Initially, it fetches all the instances of Users Services and Timelines Services from routes.cfg (Includes the urls of instances of Users services and timelines services)

For example: Here, in this project, we are running 3 instances of Users Service and 3 instances of Timelines Service

2. Gateway acts a Load Balancer by distributing the load into all of the above instances.
3. Gateway also performs **HTTP Basic Authentication** - user authentication(calls authenticate user of app.py as in Project 2) – If the user authentication is successful, then it delegate the requests to the appropriate service, else authentication failed message is displayed to the user.

4. Changes in Users Microservice

ALL USERNAMES AND PASSWORDS ARE CASE SENSITIVE.

File Name: app.py

NOTE: Users Microservice Runs in 5100, 5101, 5102 PORTS

- All the URL ENDPOINTS in Users Microservice are updated to differentiate the calls between users service or timelines service.
- We need to pass Username and Password for each request to authenticate the user.
- Please find the below changes for each of the URL's.

2.1 createUser API:

API Endpoint:	/v1users/createUser
Method:	POST
Request Content Type:	Application/JSON
Response Content Type:	Application/JSON

Working:

Send Request in the below format:

```
curl -u Pranathi:Pass@123 localhost:5000/v1users/createUser -d '{"username": "Pranathi", "password": "Pass@123", "email": "ivpranathi@csu.fullerton.edu"}' -H 'Content-Type: application/json'
```

Screenshot demonstrating the working of createUser – and the response received

```
student@tuffix-vm:~/Desktop/Project4/Project4_Final$ curl -u Julia:Pass@123 localhost:5000/v1users/createUser -d '{"username": "Julia", "password": "Pass@123", "email": "ivpranathi@csu.fullerton.edu"}' -H 'Content-Type: application/json'
```

```
{
  "ContentLanguage": "en-US",
  "ContentType": "application/json",
  "Message": "User Created Successfully",
  "status_code": 200
}
```

2.2 authenticateUser API:

API Endpoint:	/v1users/authenticateUser
Method:	POST
Request Content Type:	Application/JSON
Response Content Type:	Application/JSON

Working:

Send Request in the below format:

```
curl -u Pranathi:Pass@123 localhost:5000/v1users/authenticateUser
```

Screenshot demonstrating the working of Authenticate User – Success Response

```
student@tuffix-vm:~/Desktop/Project4/Project4_Final$ curl -u Pranathi:Pass@123 localhost:5000/v1users/authenticateUser
{
  "ContentLanguage": "en-US",
  "ContentType": "application/json",
  "Message": true,
  "status_code": 200
}
```

2.3 addFollower API:

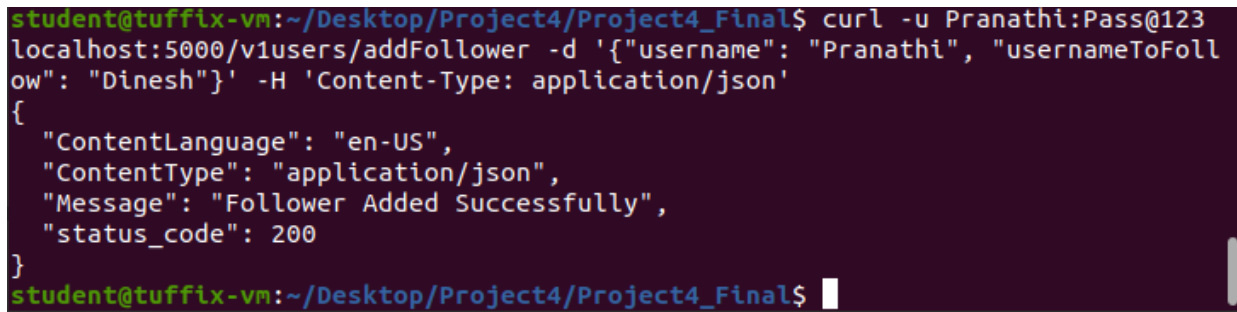
API Endpoint:	/v1users/addFollower
Method:	POST
Request Content Type:	Application/JSON
Response Content Type:	Application/JSON

Working:

Send Request in the below format:

```
curl -u Pranathi:Pass@123 localhost:5000/v1users/addFollower -d '{"username": "Pranathi", "usernameToFollow": "Dinesh"}' -H 'Content-Type: application/json'
```

Screenshot demonstrating the working of addFollower—[Success Response](#)



```
student@tuffix-vm:~/Desktop/Project4/Project4_Final$ curl -u Pranathi:Pass@123 localhost:5000/v1users/addFollower -d '{"username": "Pranathi", "usernameToFollow": "Dinesh"}' -H 'Content-Type: application/json'
{
  "ContentLanguage": "en-US",
  "ContentType": "application/json",
  "Message": "Follower Added Successfully",
  "status_code": 200
}
student@tuffix-vm:~/Desktop/Project4/Project4_Final$
```

2.4 removeFollower API:

removeFollower API enables the user to stop following a particular user.

API Endpoint:	/v1users/removeFollower
Method:	POST
Request Content Type:	Application/JSON
Response Content Type:	Application/JSON

Working:

Send Request in the below format:

```
curl -u Pranathi:Pass@123 localhost:5000/v1users/removeFollower -d '{"username": "Pranathi", "usernameToRemove": "Dinesh"}' -H 'Content-Type: application/json'
```

Screenshot demonstrating the working of removeFollower- Success Response

```
student@tufflx-vm:~/Desktop/Project4/Project4_Final$ curl -u Pranathi:Pass@123
localhost:5000/v1users/removeFollower -i -X DELETE -H 'Content-Type: applicat
ion/json' -d '{"username": "Pranathi", "usernameToRemove": "Dinesh"}'
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 141
Server: Werkzeug/1.0.1 Python/3.8.5
Date: Fri, 25 Dec 2020 04:54:12 GMT

{
  "ContentLanguage": "en-US",
  "ContentType": "application/json",
  "Message": "Follower Removed Successfully",
  "status_code": 200
}
```

5 Changes in Timeline Microservice

ALL USERNAMES AND PASSWORDS ARE CASE SENSITIVE.

File Name: timelinesApi.py

NOTE: Users Microservice Runs in 5200, 5201, 5202 PORTS

- All the URL ENDPOINTS in Timelines Microservice are updated to differentiate the calls between users service or timelines service.
- We need to pass Username and Password for each request to authenticate the user.
- Please find the below changes for each of the URL's.

3.1 postTweet API:

postTweet API

API Endpoint:	/v1timelines/postTweet
Method:	POST
Request Content Type:	Application/JSON
Response Content Type:	Application/JSON

Working:

Send Request in the below format:

```
curl -u Pranathi:Pass@123 localhost:5000/v1timelines/postTweet -d '{"username": "Pranathi", "post": "Its been a great day!!"}' -H 'Content-Type: application/json'
```

Screenshot demonstrating the working of postTweet– Success Response

```
student@tufflx-vm:~/Desktop/Project4/Project4_Final$ curl -u Pranathi:Pass@123 localhost:5000/v1timelines/postTweet -d '{"username": "Pranathi", "post": "Its been a great day!!"}' -H 'Content-Type: application/json'
{
  "ContentLanguage": "en-US",
  "ContentType": "application/json",
  "Message": "Tweet Posted Successfully",
  "StatusCode": 200
}
```

3.2 getUserTimeline API:

- This api is updated to receive query parameters username and password in the below format as shown in working.
- Here the tweets of the particular user is termed as searchUsername in the API

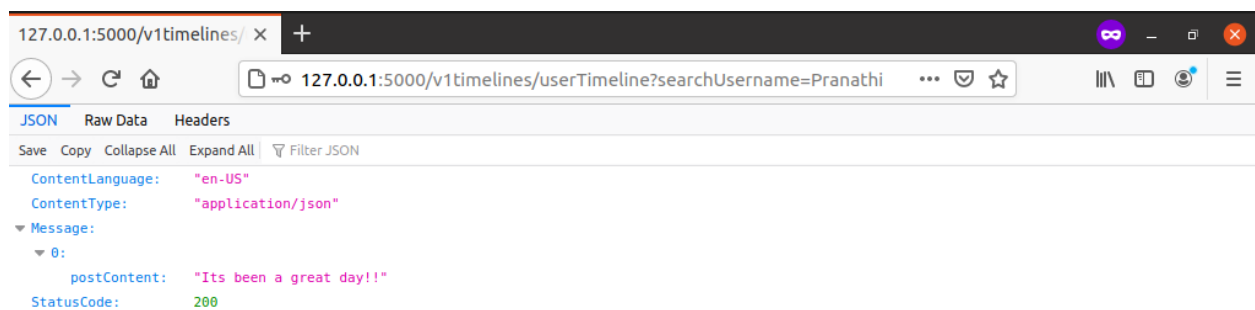
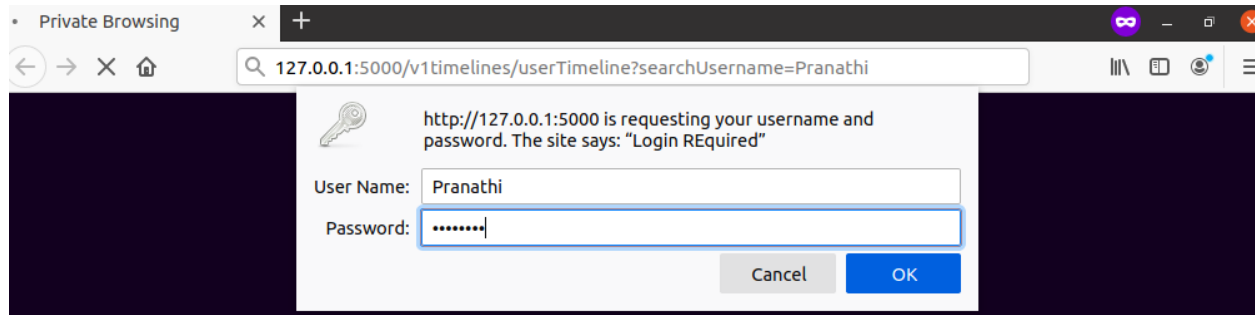
API Endpoint:	/v1timelines/userTimeline
Method:	GET
Response Content Type:	Application/JSON

Working:

Send Request in the below format:

<http://127.0.0.1:5000/v1timelines/userTimeline?searchUsername=Pranathi>

Screenshot demonstrating the working of getUserTimeline – Success Response



3.3 getPublicTimeline API:

getPublicTimeline_API

API Endpoint: /v1timelines/publicTimeline

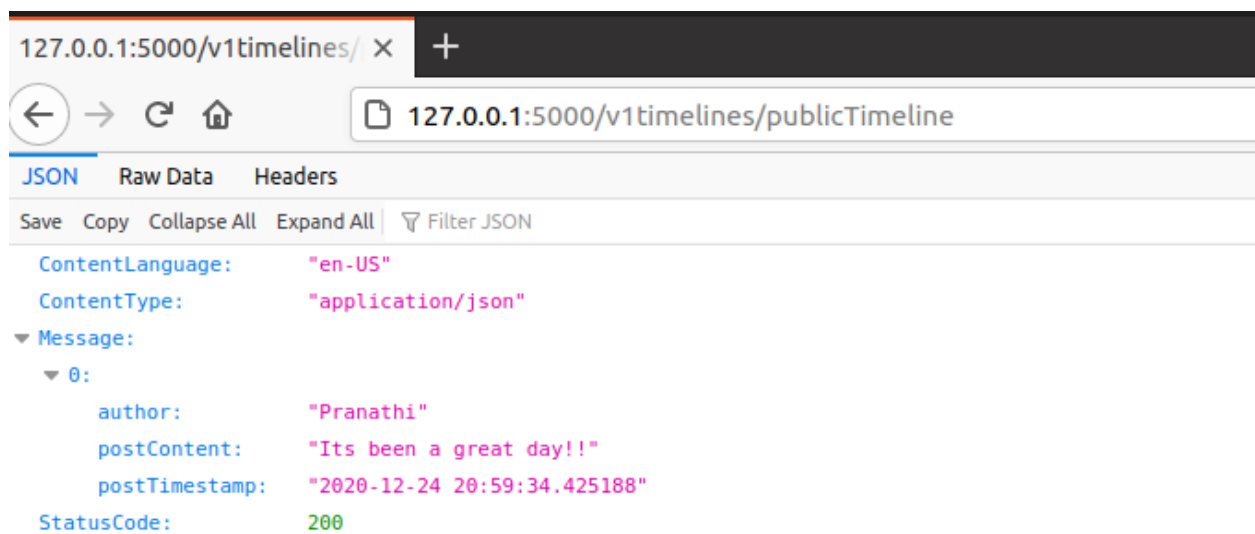
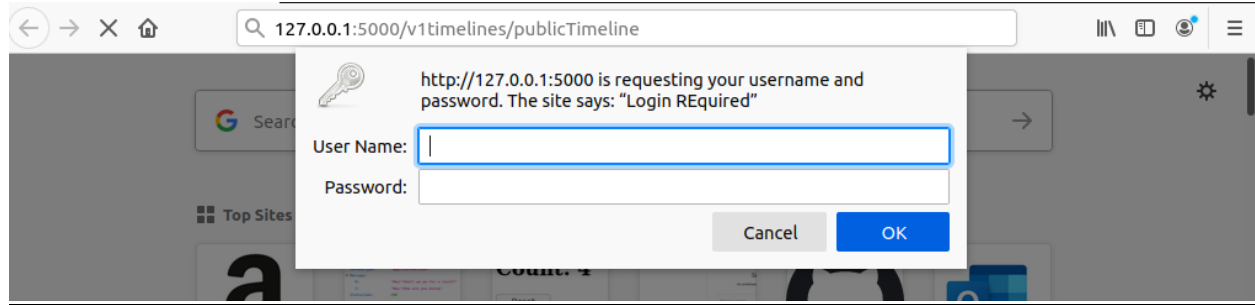
Method: GET

Response Content Type: Application/JSON

Send Request in the below format:

<http://127.0.0.1:5000/v1timelines/publicTimeline>

Screenshot demonstrating the working of getPublicTimeline – Success Response



3.4 getHomeTimeline API:

getHomeTimeline_API Returns recent 25 tweets from all the users, that this specified user follows.

API Endpoint: /v1/homeTimeline

Method: GET

Response Content Type: Application/JSON

Working:

Send Request in the below format:

`http://127.0.0.1:5000/v1timelines/homeTimeline?username=Pranathi`

Screenshot demonstrating the working of getHomeTimeline – Success Response

