CPSC 476

Assignment 3

Database Tables: Schemas of required tables used by our application.

Table 1: links

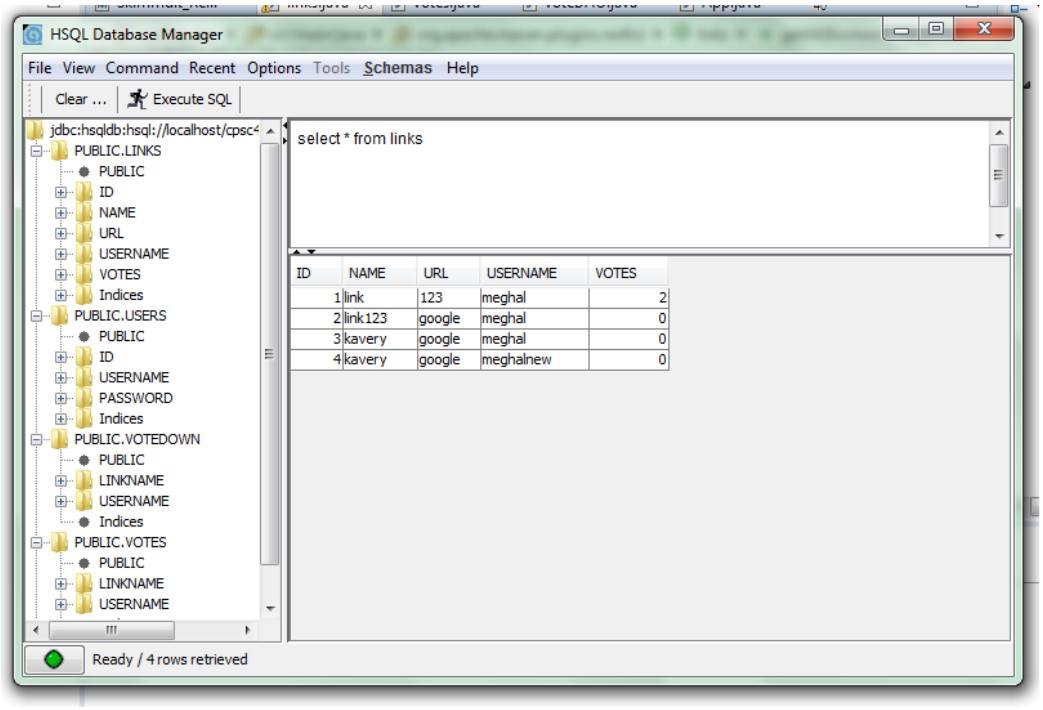


Table 2: users

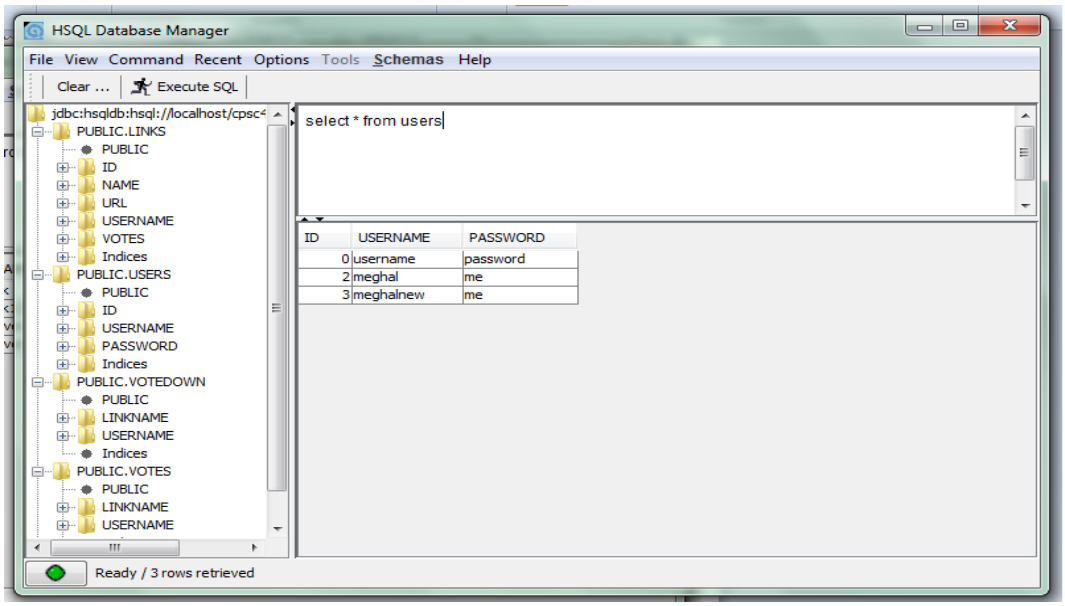


Table 3: votedown

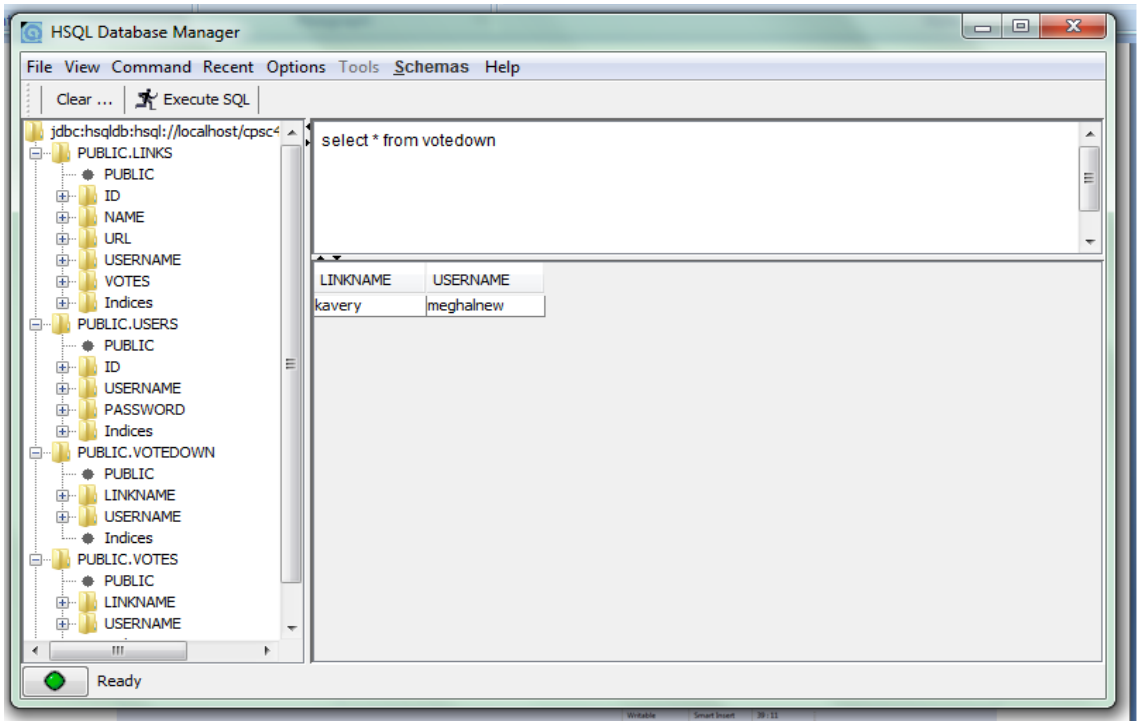
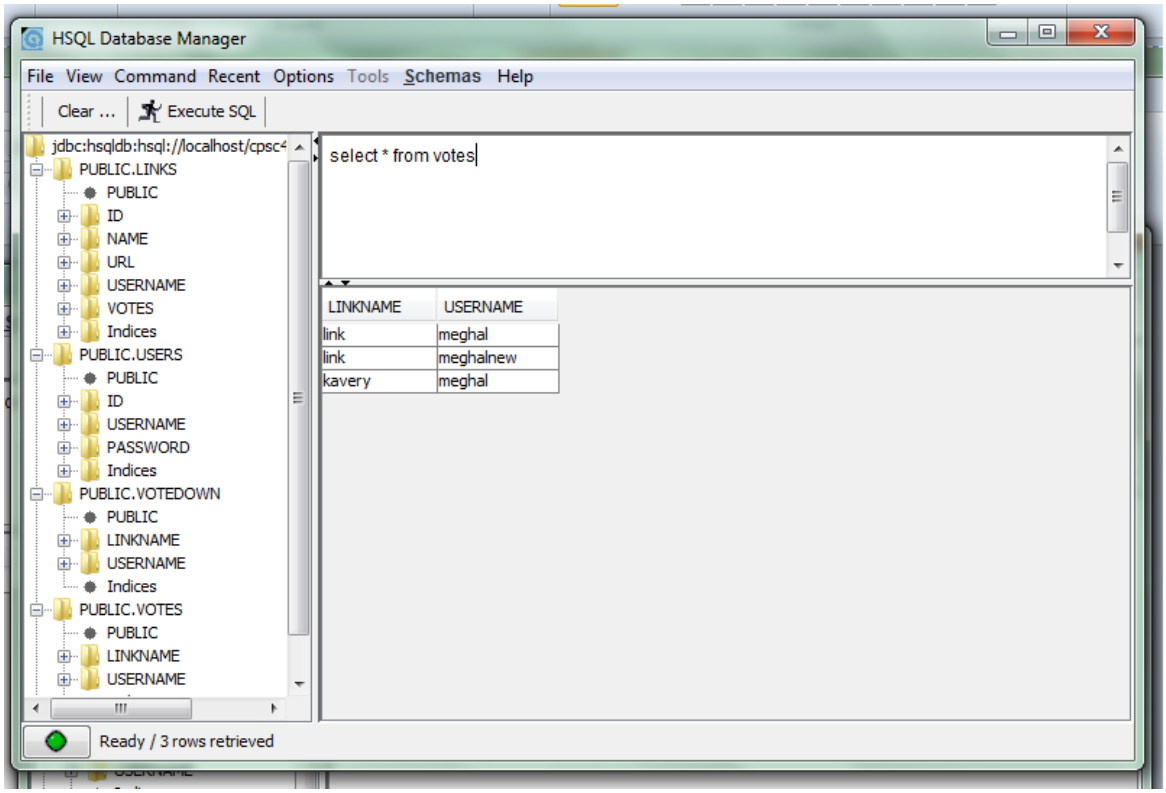


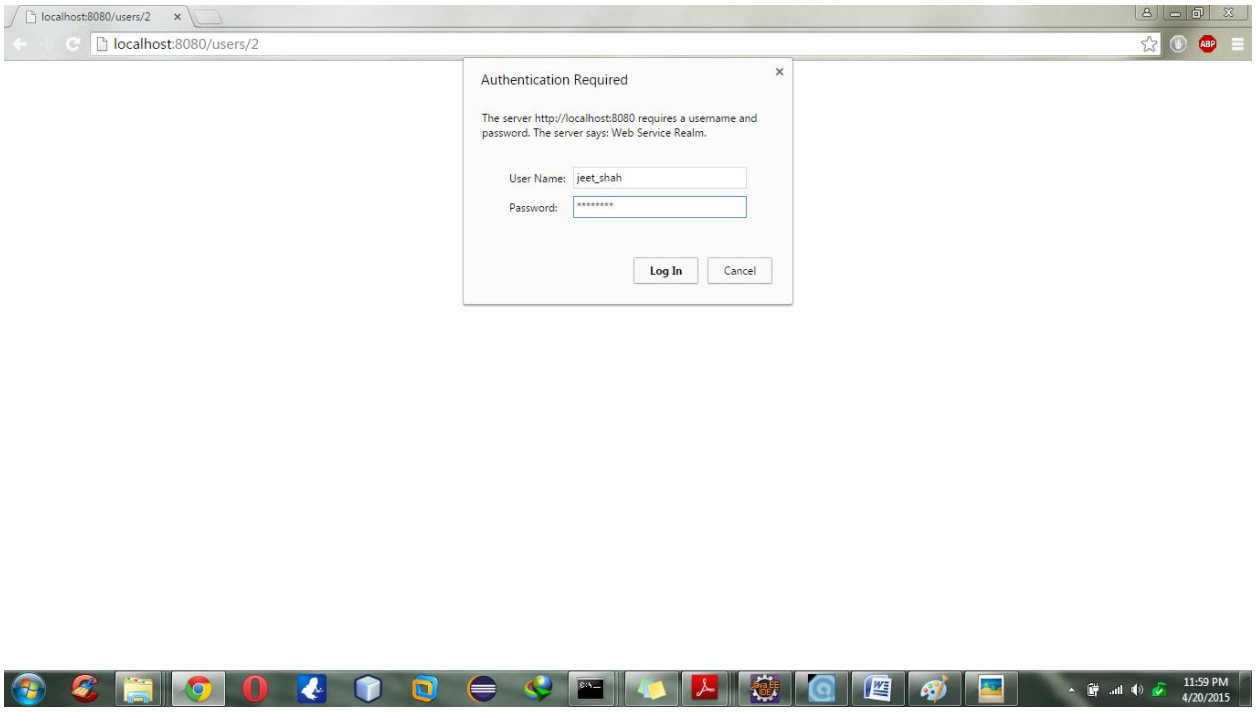
Table 4: votes



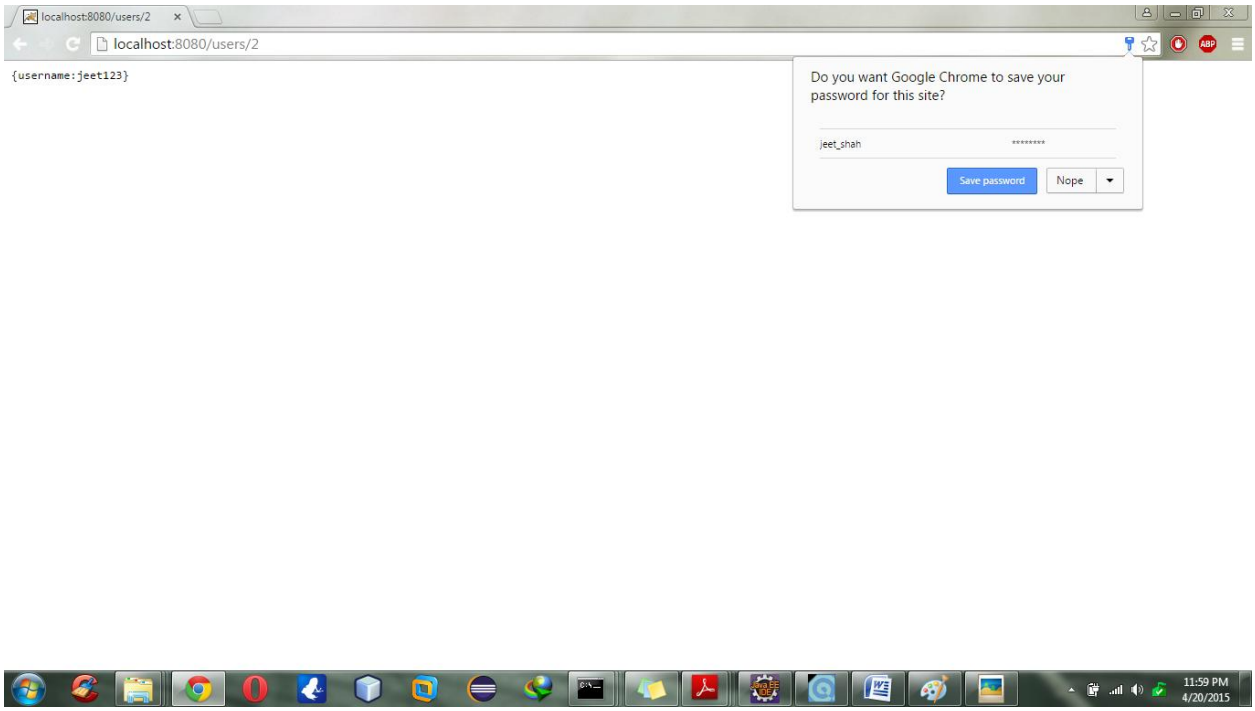
Resources: (For Http basic authentication, username: jeet_shah and password: mememe92)

➔ As per requirement, we have 3 resources: users, links, and votes.
(1) Users:

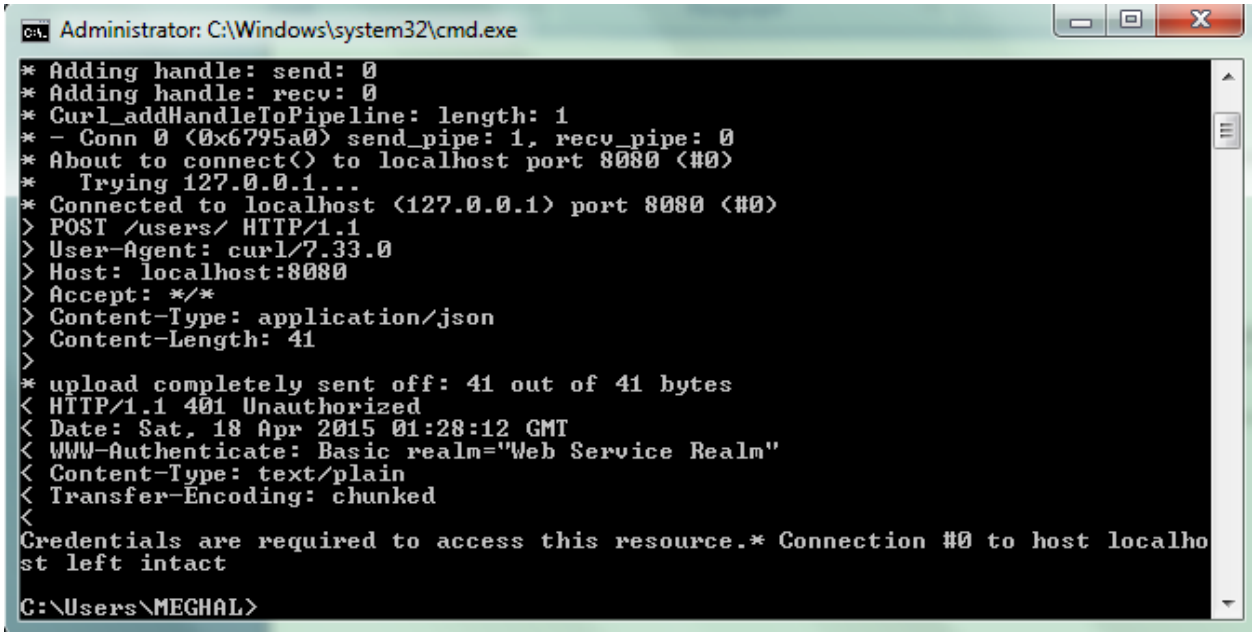
1) 'Get' method on user resource.
→ When we try to 'get' the user resource, http basic authentication window is popped up. We entered the correct user name and password.



→ After giving the correct credentials (username: jeet_shah and password: mememe92), we got the user resource in json form.



2) POST method on user resource



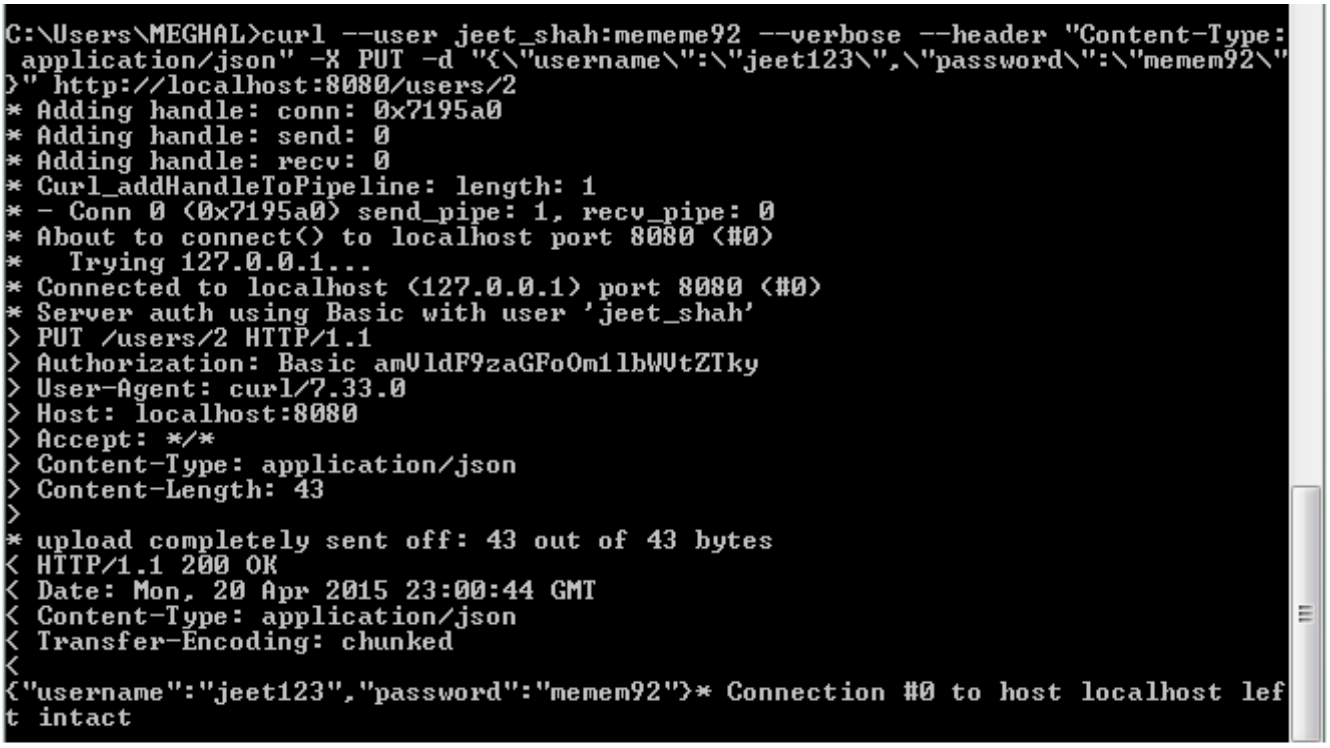
→Here, we post the request. But, http basic authentication refuses because user credentials are not correct.

```
C:\Users\MEGHAL>curl --user jeet_shah:mememe92 --verbose --header "Content-Type:
 application/json" -X POST -d "{\"username\":\"jeet\",\"password\":\"memem92\"}"
 http://localhost:8080/users
* Adding handle: conn: 0x6895a0
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x6895a0) send_pipe: 1, recv_pipe: 0
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 8080 (#0)
* Server auth using Basic with user 'jeet_shah'
> POST /users HTTP/1.1
> Authorization: Basic amVldF9zaGFoOm1lbWVtZTky
> User-Agent: curl/7.33.0
> Host: localhost:8080
> Accept: */*
> Content-Type: application/json
> Content-Length: 40
>
* upload completely sent off: 40 out of 40 bytes
< HTTP/1.1 201 Created
< Date: Mon, 20 Apr 2015 22:59:22 GMT
< Location: http://localhost:8080/users/4
< Content-Type: application/json
< Content-Length: 0
<
* Connection #0 to host localhost left intact

C:\Users\MEGHAL>
```

→Now, we again 'post' the request on user resource with correct username and password (http basic authentication). This time request is processed successfully.

3) PUT method on user resource



→ We 'put' the request (update) on the user resource with the required user credentials via http basic authentication. We can see that it was processed successfully and we update the json resource.

4) DELETE method on user resource

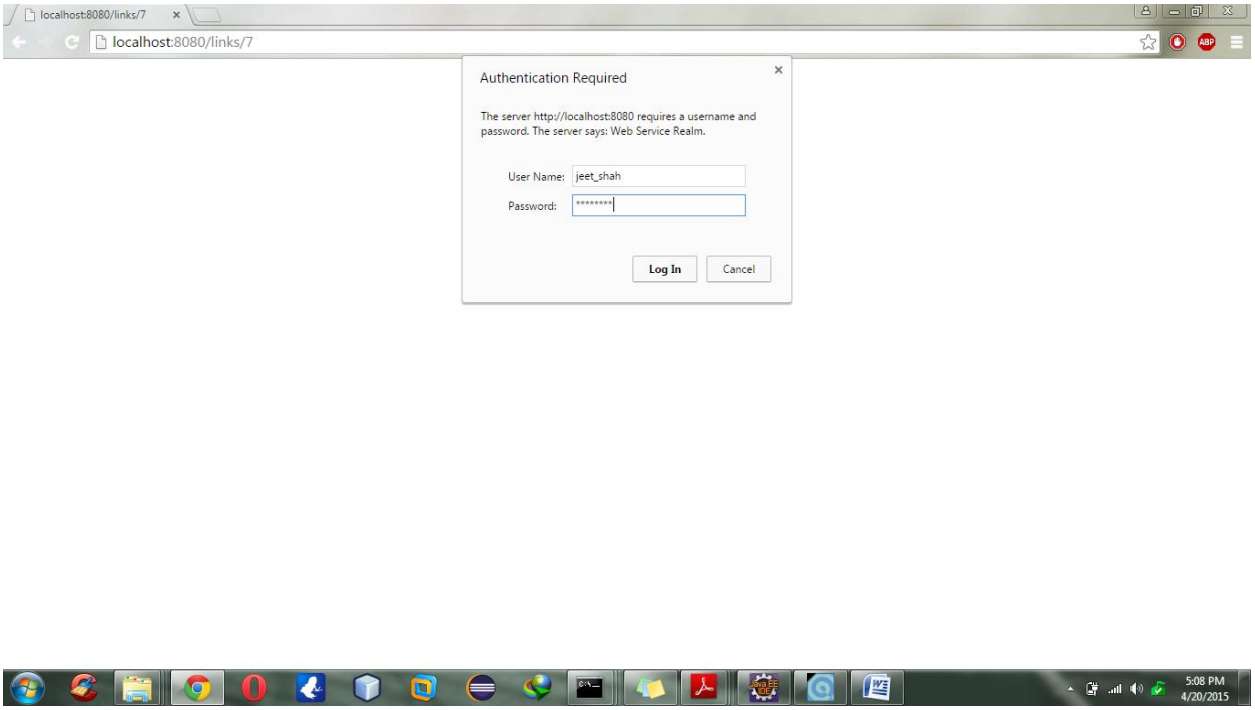
→ First, we try to delete user resource without giving user credentials but it won't allow. Then, we have applied correct user credentials (http basic authentication) and then user was deleted.

➔ **NOTE:** All these requests successfully reflect respective details into database, after successfully completion of request.

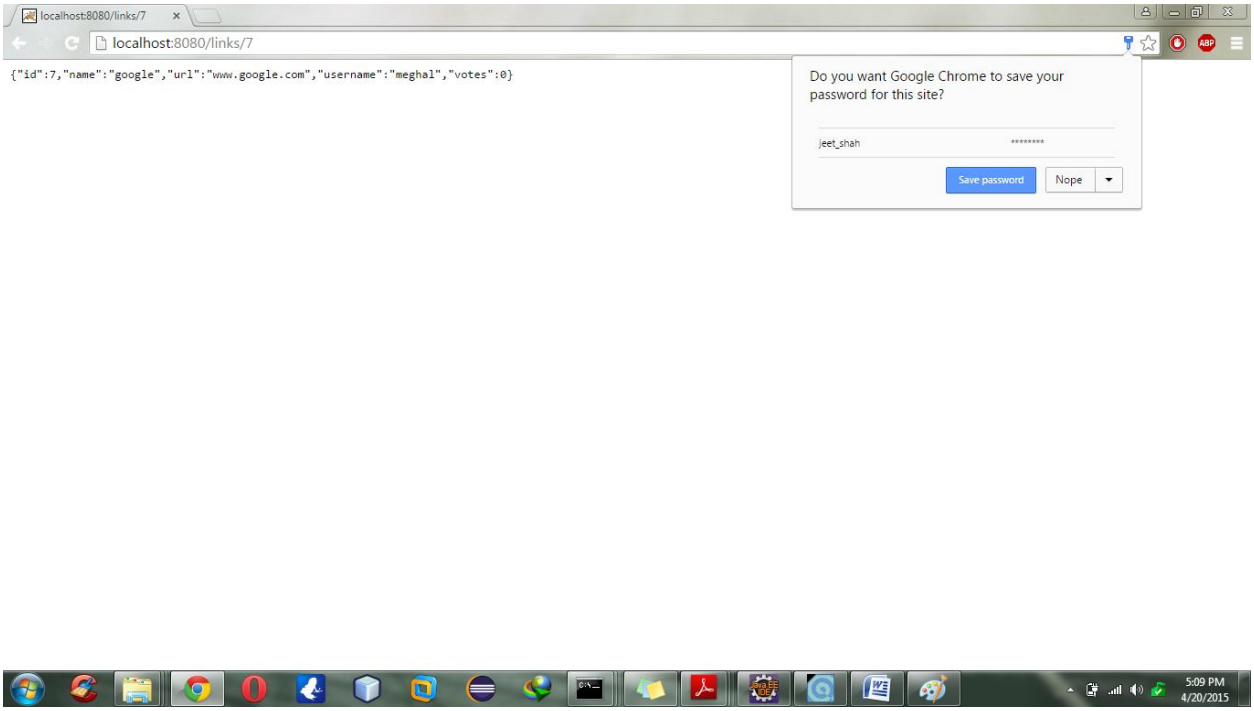
2) Links

NOTE: Http basic authentication is same for other two links and votes resources, so we didn't show it here in browser as we have already showed it for 'users'. But, we have showed them in curl.

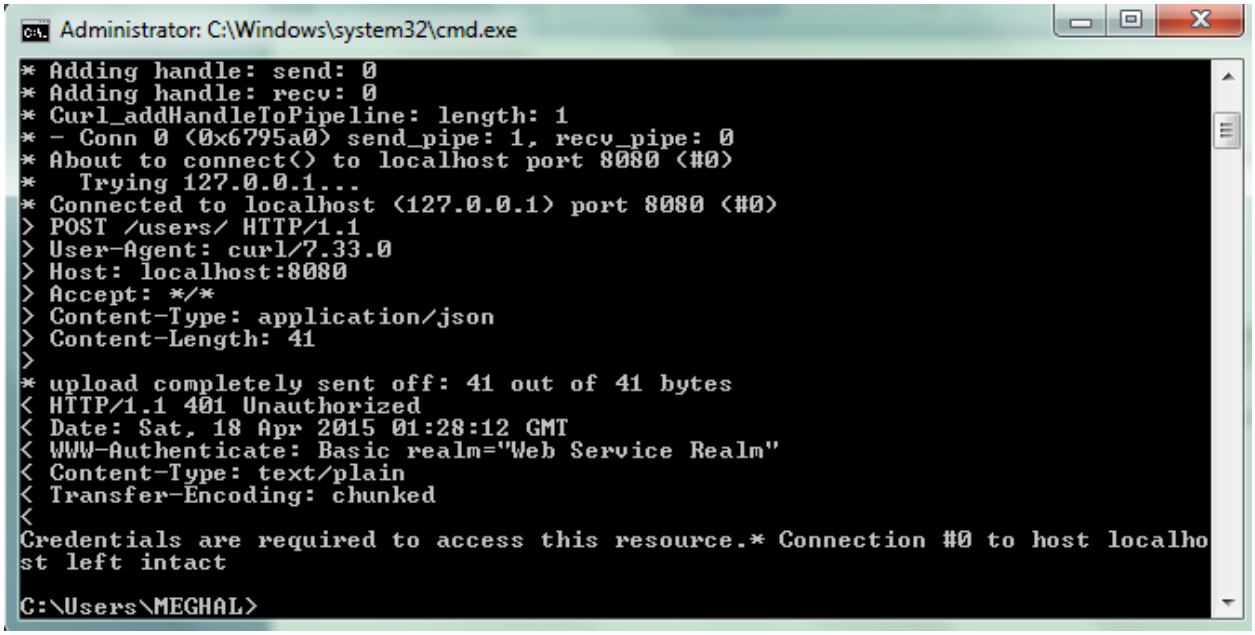
1) GET method on link resource
   → When we try to 'get' the links resource, http basic authentication window is popped up. We entered the correct user name and password.



→ After giving the correct credentials (username: jeet_shah and password: mememe92), we got the user resource in json form.



2) POST method on link resource



→Here, we post the request. But, http basic authentication refuses because user credentials are not correct.

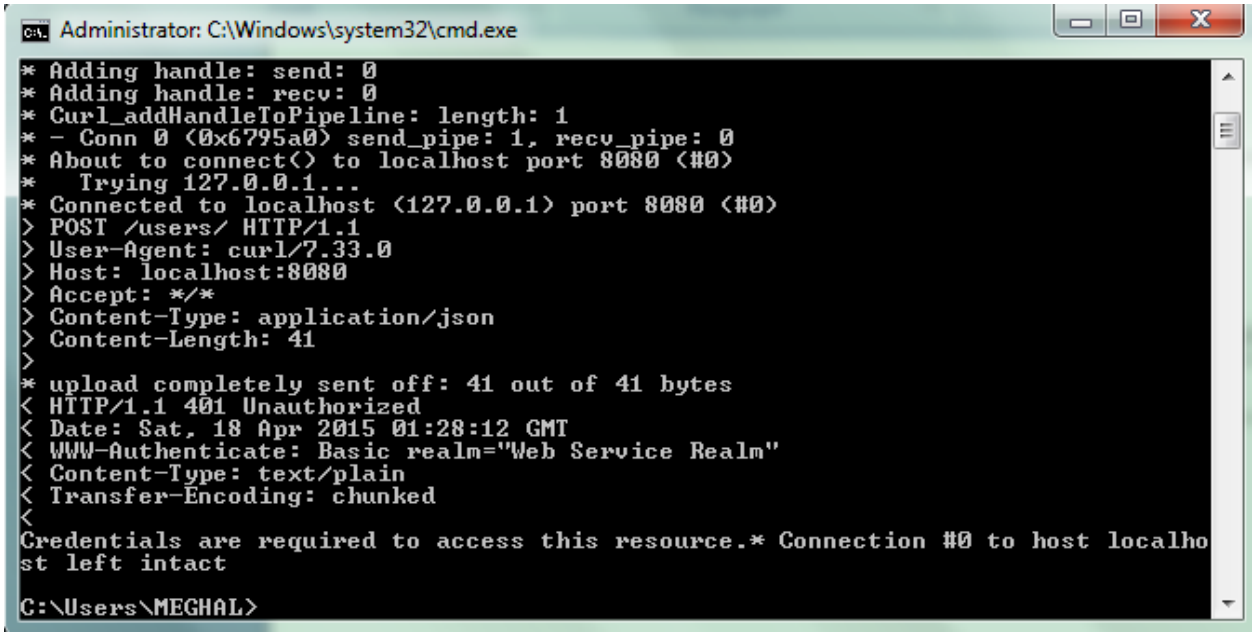
```
* upload completely sent off: 70 out of 70 bytes
< HTTP/1.1 401 Unauthorized
< Date: Mon, 20 Apr 2015 22:22:35 GMT
< WWW-Authenticate: Basic realm="Web Service Realm"
< Content-Type: text/plain
< Transfer-Encoding: chunked
<
Credentials are required to access this resource.* Connection #0 to host localho
st left intact

C:\Users\MEGHAL>curl --user jeet_shah:mememe92 --verbose --header "Content-Type:
 application/json" -X POST -d "{\"name\":\"google\",\"url\":\"www.google.com\",\
"username\":\"meghal\",\"votes\":0}" http://localhost:8080/links
* Adding handle: conn: 0x5d95a0
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x5d95a0) send_pipe: 1, recv_pipe: 0
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 8080 (#0)
* Server auth using Basic with user 'jeet_shah'
> POST /links HTTP/1.1
> Authorization: Basic amVldF9zaGFoOm1lbWVtZTky
> User-Agent: curl/7.33.0
> Host: localhost:8080
> Accept: */*
> Content-Type: application/json
> Content-Length: 70
>
* upload completely sent off: 70 out of 70 bytes
< HTTP/1.1 201 Created
< Date: Mon, 20 Apr 2015 22:22:58 GMT
< Location: http://localhost:8080/links/5
< Content-Type: application/json
< Content-Length: 0
<
* Connection #0 to host localhost left intact

C:\Users\MEGHAL>_
```

→Now, we again 'post' the request on user resource with correct username and password (http basic authentication). This time request is processed successfully.

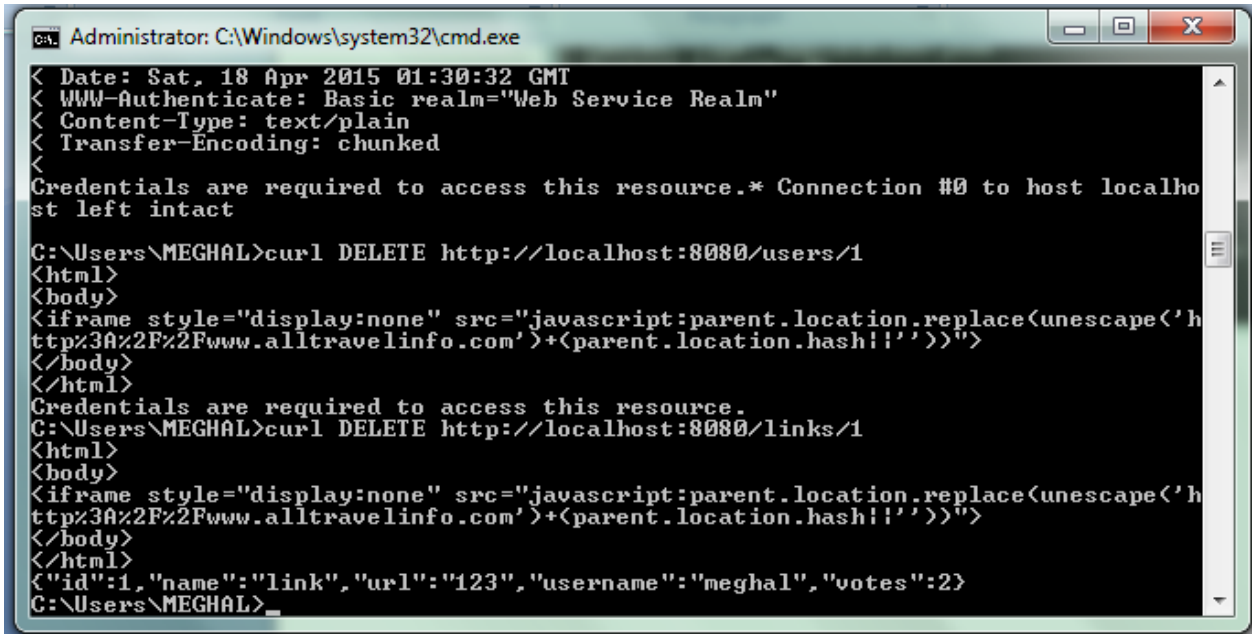
3)PUT method on link resource



→ We ‘put’ the request (update) on the link resource with the required user credentials via http basic authentication. We can see that it is processed successfully and we update the json resource.

```
C:\Users\MEGHAL>curl --user jeet_shah:mememe92 --verbose --header "Content-Type:
 application/json" -X PUT -d "{\"name\":\"google\",\"url\":\"www.google.com\",\"
username\":\"meghal\",\"votes\":0}" http://localhost:8080/links/1
* Adding handle: conn: 0x7d95a0
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x7d95a0) send_pipe: 1, recv_pipe: 0
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 8080 (#0)
* Server auth using Basic with user 'jeet_shah'
> PUT /links/1 HTTP/1.1
> Authorization: Basic amVldF9zaGFoOm1lbWVtZTky
> User-Agent: curl/7.33.0
> Host: localhost:8080
> Accept: */*
> Content-Type: application/json
> Content-Length: 70
>
* upload completely sent off: 70 out of 70 bytes
< HTTP/1.1 200 OK
< Date: Mon, 20 Apr 2015 22:31:34 GMT
< Content-Type: application/json
< Transfer-Encoding: chunked
<
{"id":1,"name":"google","url":"www.google.com","username":"meghal","votes":0}* C
onnection #0 to host localhost left intact
```

4) DELETE method on link resource



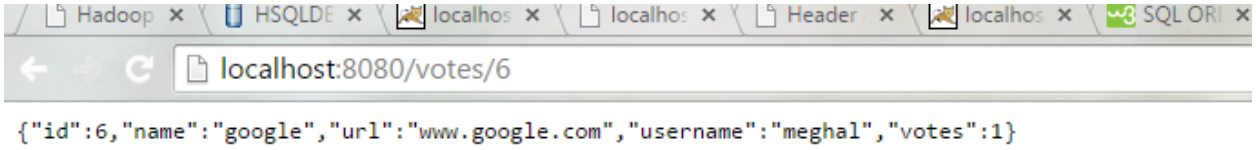
→Here, we try delete request. But, http basic authentication refuses because user credentials are not correct.

→After giving the valid user credentials, delete request processes successfully. We can see it in the message.

3) Votes:

NOTE: Http basic authentication is same for all three resources, so we didn't show it here in **browser** as we have already showed it for 'users'. But, we have showed it in curl as we have implemented it.

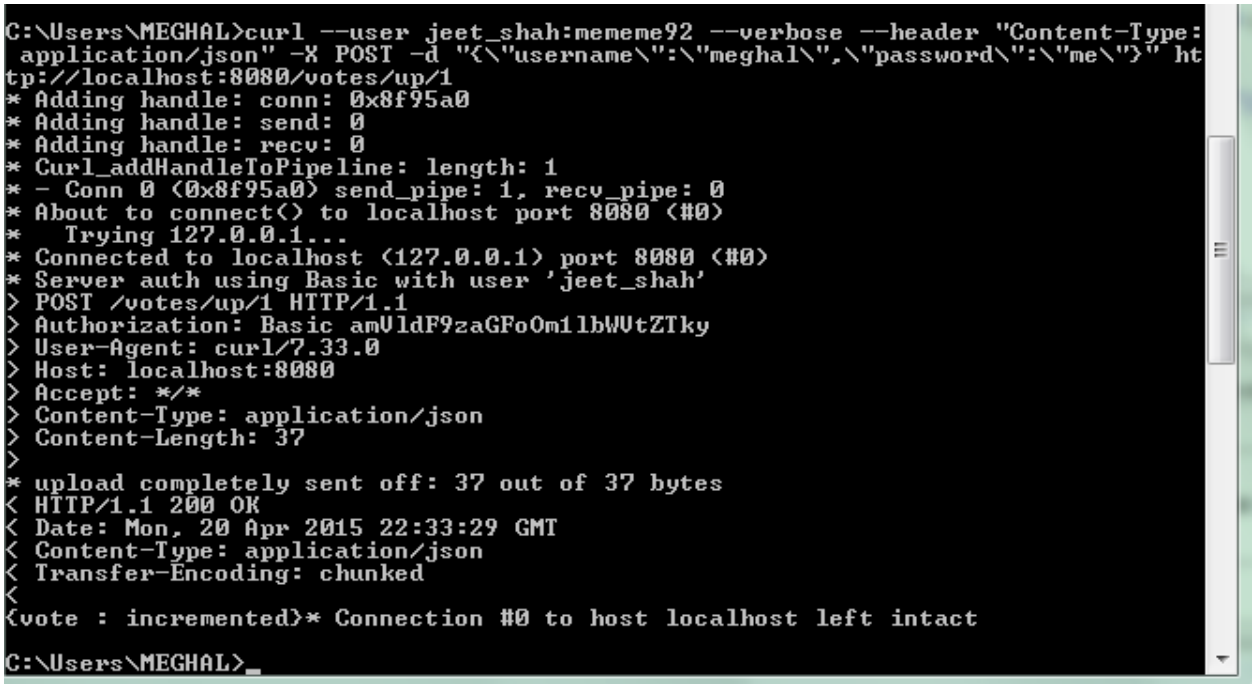
1) Get request on votes resource:



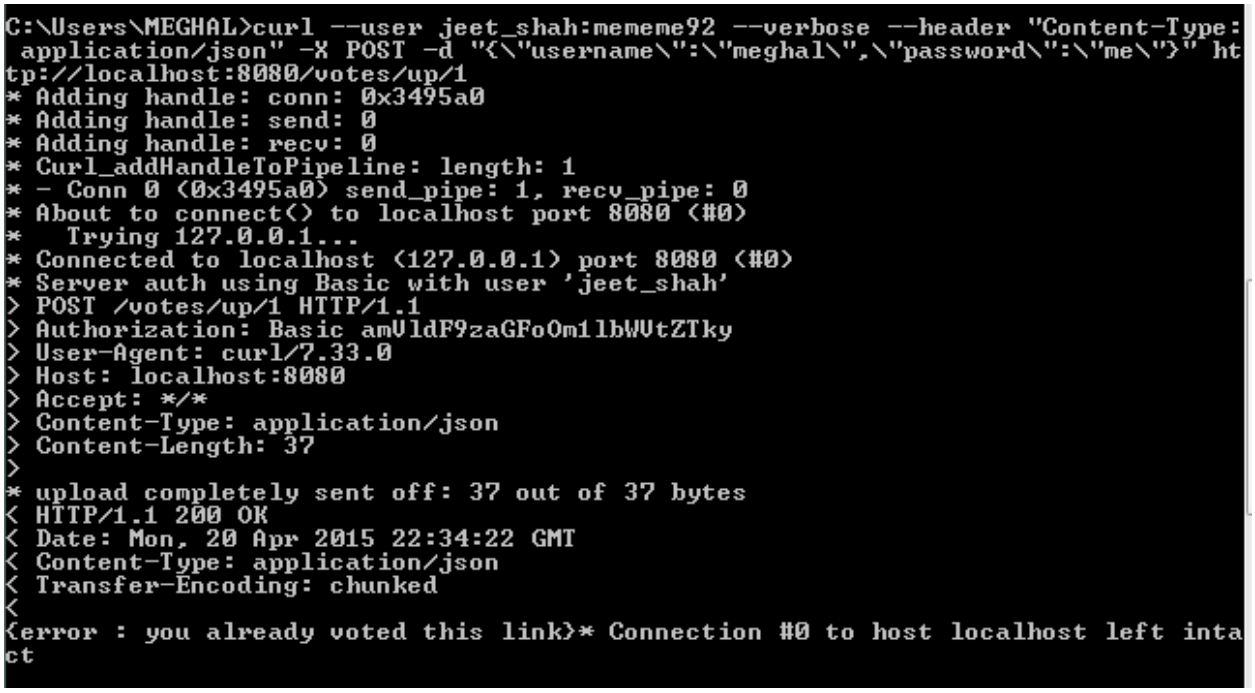
→When we try to get the sixth link via above url, Json response is returned. We can see all the required details of this link

2) Post request on votes resource

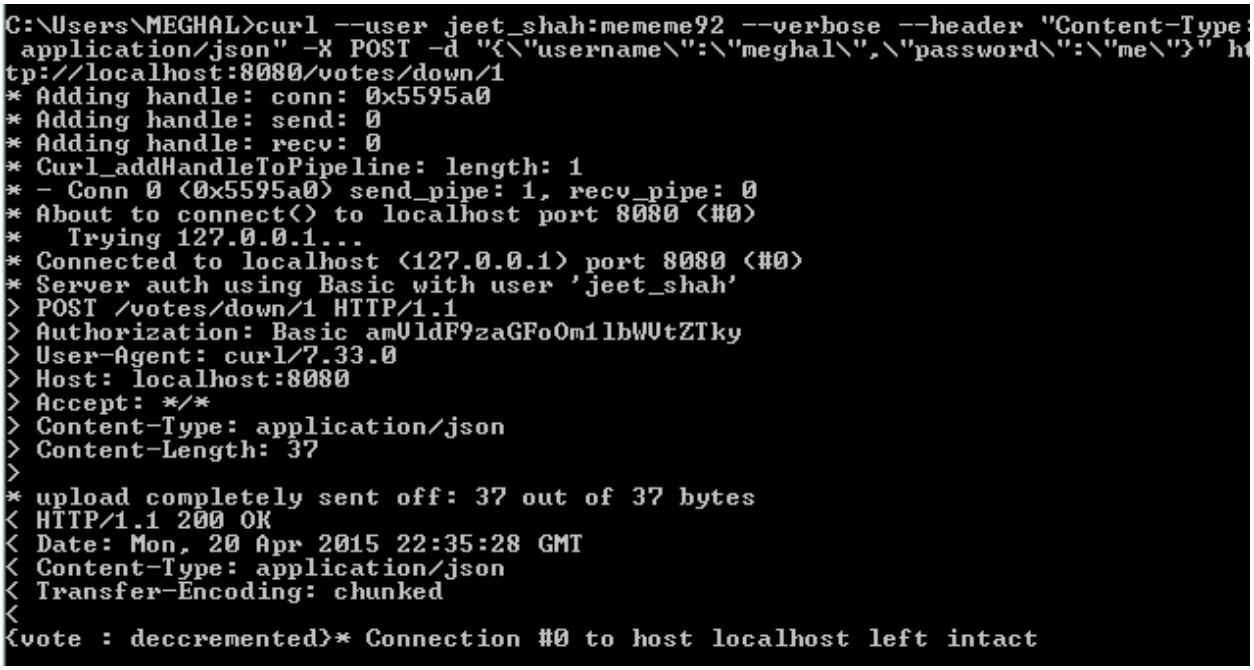
→If user wants **to like the link** (to increment the vote), then we use this url: http://localhost:8080/votes/up/1 and give the required json representation (user and password) along with it. Thus, link that has id=1, its vote is incremented by 1. We have also written the successful message in the end. These particular details are reflected in database, too (obviously). (You can check this in database tables showed in beginning of this documentation.)



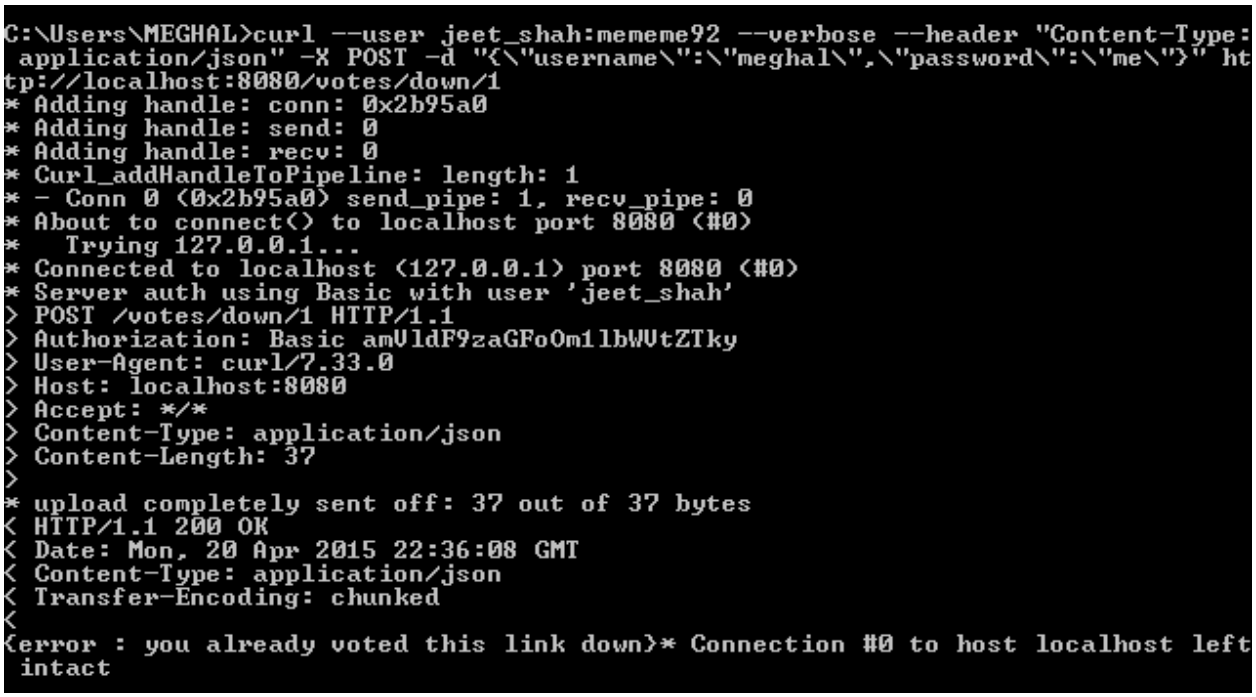
→After giving the vote, if user tries to vote the same link again (here id=1), then it denied and we have printed the error message as a notification for the user.



→If user wants to unlike the link (to decrement the vote), then we use this url: http://localhost:8080/votes/down/1 and give the required json representation (user and password) along with it. Thus, link that has id=1, its vote is decremented by 1. We have also written the successful message in the end. These particular details are reflected in database, too (obviously). (You can check this in database tables showed in beginning of this documentation.)



→After unlike the link, if user tries to unlike (vote down) the same link again (here id=1), then it denied and we have printed the error message as a notification for the user.



→Here, we have showed all the links in GET request. It doesn't require the user credentials (http basic authentication) because anyone can see all links. We have sorted it according to votes of respective link (see the last field 'votes').