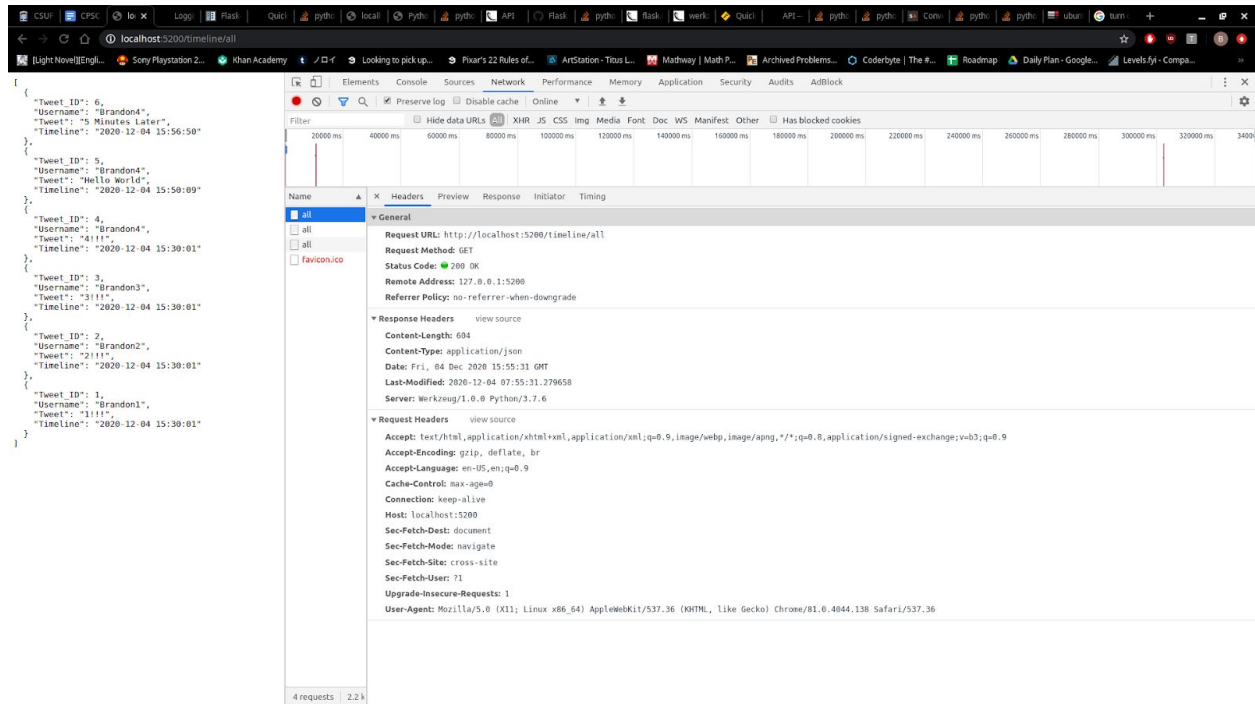


Screenshots

Test HTTP caching using your Browser's Developer Tools to verify that:

1. The initial call to `getPublicTimeline()` loads JSON from the server



2. Refreshing the page within a 5 minute window results in HTTP 304 Not Modified with no payload.

The screenshot shows a web browser window with a REST client on the left and a timeline of API requests on the right. The REST client shows a sequence of requests for a timeline feed, with the first request returning a 304 status code. The timeline shows a sequence of requests for a timeline feed, with the first request returning a 304 status code.

```
{
  "Tweet_ID": 6,
  "Username": "Brandon4",
  "Tweet": "5 Minutes Later",
  "Timeline": "2020-12-04 15:56:50"
},
{
  "Tweet_ID": 5,
  "Username": "Brandon4",
  "Tweet": "Hello World",
  "Timeline": "2020-12-04 15:58:09"
},
{
  "Tweet_ID": 4,
  "Username": "Brandon4",
  "Tweet": "4!!!",
  "Timeline": "2020-12-04 15:38:01"
},
{
  "Tweet_ID": 3,
  "Username": "Brandon3",
  "Tweet": "3!!!",
  "Timeline": "2020-12-04 15:38:01"
},
{
  "Tweet_ID": 2,
  "Username": "Brandon2",
  "Tweet": "2!!!",
  "Timeline": "2020-12-04 15:38:01"
},
{
  "Tweet_ID": 1,
  "Username": "Brandon1",
  "Tweet": "1!!!",
  "Timeline": "2020-12-04 15:38:01"
}
}
```

4 requests 2.2s

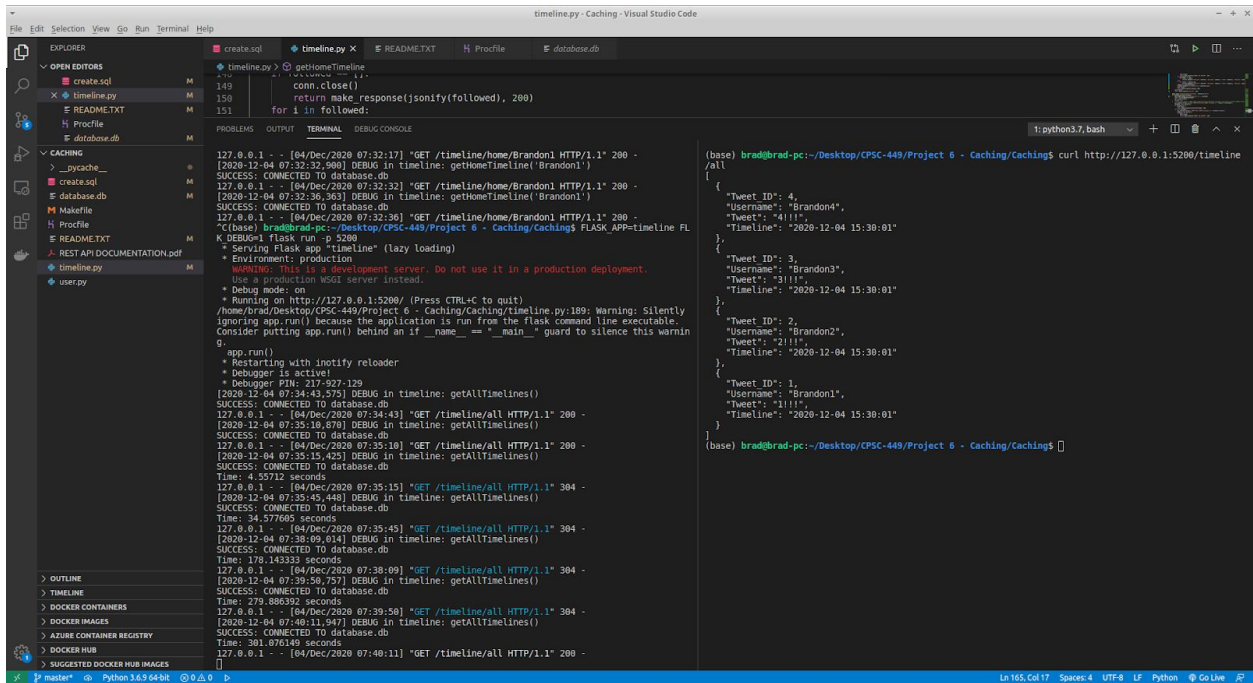
3. Calls outside the 5 minute window reload the JSON for the timeline.

The screenshot shows a web browser window with a REST client on the left and a timeline of API requests on the right. The REST client shows a sequence of requests for a timeline feed, with the first request returning a 200 status code. The timeline shows a sequence of requests for a timeline feed, with the first request returning a 200 status code.

```
{
  "Tweet_ID": 6,
  "Username": "Brandon4",
  "Tweet": "5 Minutes Later",
  "Timeline": "2020-12-04 15:56:50"
},
{
  "Tweet_ID": 5,
  "Username": "Brandon4",
  "Tweet": "Hello World",
  "Timeline": "2020-12-04 15:58:09"
},
{
  "Tweet_ID": 4,
  "Username": "Brandon4",
  "Tweet": "4!!!",
  "Timeline": "2020-12-04 15:38:01"
},
{
  "Tweet_ID": 3,
  "Username": "Brandon3",
  "Tweet": "3!!!",
  "Timeline": "2020-12-04 15:38:01"
},
{
  "Tweet_ID": 2,
  "Username": "Brandon2",
  "Tweet": "2!!!",
  "Timeline": "2020-12-04 15:38:01"
},
{
  "Tweet_ID": 1,
  "Username": "Brandon1",
  "Tweet": "1!!!",
  "Timeline": "2020-12-04 15:38:01"
}
}
```

4 requests 2.2s

4. Logged Screenshots



```
create.sql | timeline.py | README.txt | Profile | database.db
EXPLORER
OPEN EDITORS
create.sql M
timeline.py X
README.txt M
Profile M
database.db M
CACHING
_pycache_ M
create.sql M
database.db M
Makefile M
README.txt M
REST API DOCUMENTATION.pdf M
timeline.py M
user.py M
timeline.py > getHomeTimeline
149
150
151
return jsonify(followed), 200)
for i in followed:
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
127.0.0.1 - - [04/Dec/2020 07:32:17] "GET /timeline/home/Brandon1 HTTP/1.1" 200 -
[2020-12-04 07:32:32.908] DEBUG in timeline: getHomeTimeline('Brandon1')
SUCCESS: CONNECTED TO database.db
127.0.0.1 - - [04/Dec/2020 07:32:32] "GET /timeline/home/Brandon1 HTTP/1.1" 200 -
[2020-12-04 07:32:36.363] DEBUG in timeline: getHomeTimeline('Brandon1')
SUCCESS: CONNECTED TO database.db
127.0.0.1 - - [04/Dec/2020 07:32:36] "GET /timeline/home/Brandon1 HTTP/1.1" 200 -
C(base) brad@brad-pc:~/Desktop/CPSC-449/Project 6 - Caching/Caching$ FLASK_APP=timeline FL
K DEBUG=1 flask run -p 5200
* Serving Flask app "timeline" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5200/ (Press CTRL+C to quit)
/home/brad/Desktop/CPSC-449/Project 6 - Caching/Caching/timeline.py:189: Warning: Silently
ignoring app.run() because the application is run from the flask command line executable.
Consider putting app.run() behind an if __name__ == "__main__" guard to silence this warnin
g.
app.run()
* Restarting with Inotify reloader
* Debugger is active!
* Debugger PIN: 237-927-129
[2020-12-04 07:34:43.575] DEBUG in timeline: getAllTimelines()
SUCCESS: CONNECTED TO database.db
127.0.0.1 - - [04/Dec/2020 07:34:43] "GET /timeline/all HTTP/1.1" 200 -
[2020-12-04 07:35:18.670] DEBUG in timeline: getAllTimelines()
SUCCESS: CONNECTED TO database.db
127.0.0.1 - - [04/Dec/2020 07:35:18] "GET /timeline/all HTTP/1.1" 200 -
[2020-12-04 07:35:15.425] DEBUG in timeline: getAllTimelines()
SUCCESS: CONNECTED TO database.db
Time: 4.55712 seconds
127.0.0.1 - - [04/Dec/2020 07:35:15] "GET /timeline/all HTTP/1.1" 304 -
[2020-12-04 07:35:45.448] DEBUG in timeline: getAllTimelines()
SUCCESS: CONNECTED TO database.db
Time: 34.577605 seconds
127.0.0.1 - - [04/Dec/2020 07:35:45] "GET /timeline/all HTTP/1.1" 304 -
[2020-12-04 07:38:09.014] DEBUG in timeline: getAllTimelines()
SUCCESS: CONNECTED TO database.db
Time: 170.143333 seconds
127.0.0.1 - - [04/Dec/2020 07:38:09] "GET /timeline/all HTTP/1.1" 304 -
[2020-12-04 07:39:58.757] DEBUG in timeline: getAllTimelines()
SUCCESS: CONNECTED TO database.db
Time: 279.886392 seconds
127.0.0.1 - - [04/Dec/2020 07:39:58] "GET /timeline/all HTTP/1.1" 304 -
[2020-12-04 07:40:11.947] DEBUG in timeline: getAllTimelines()
SUCCESS: CONNECTED TO database.db
Time: 381.076149 seconds
127.0.0.1 - - [04/Dec/2020 07:40:11] "GET /timeline/all HTTP/1.1" 200 -
[]
(base) brad@brad-pc:~/Desktop/CPSC-449/Project 6 - Caching/Caching$ curl http://127.0.0.1:5200/timeline/all
{
  "Tweet_ID": 4,
  "Username": "Brandon4",
  "Tweet": "4!!!!",
  "Timeline": "2020-12-04 15:30:01"
},
  "Tweet_ID": 3,
  "Username": "Brandon3",
  "Tweet": "3!!!!",
  "Timeline": "2020-12-04 15:30:01"
},
  "Tweet_ID": 2,
  "Username": "Brandon2",
  "Tweet": "2!!!!",
  "Timeline": "2020-12-04 15:30:01"
},
  "Tweet_ID": 1,
  "Username": "Brandon1",
  "Tweet": "1!!!!",
  "Timeline": "2020-12-04 15:30:01"
}
(base) brad@brad-pc:~/Desktop/CPSC-449/Project 6 - Caching/Caching$
```

Object Caching

1. Using `app.logger.debug()` to determine when a request for a user's timeline is fulfilled from cache and when it results in a call to the original data source.

Logged when using original data source and when using object cache.

2. Calling `getHomeTimeline(username)` for a user and verifying its contents.

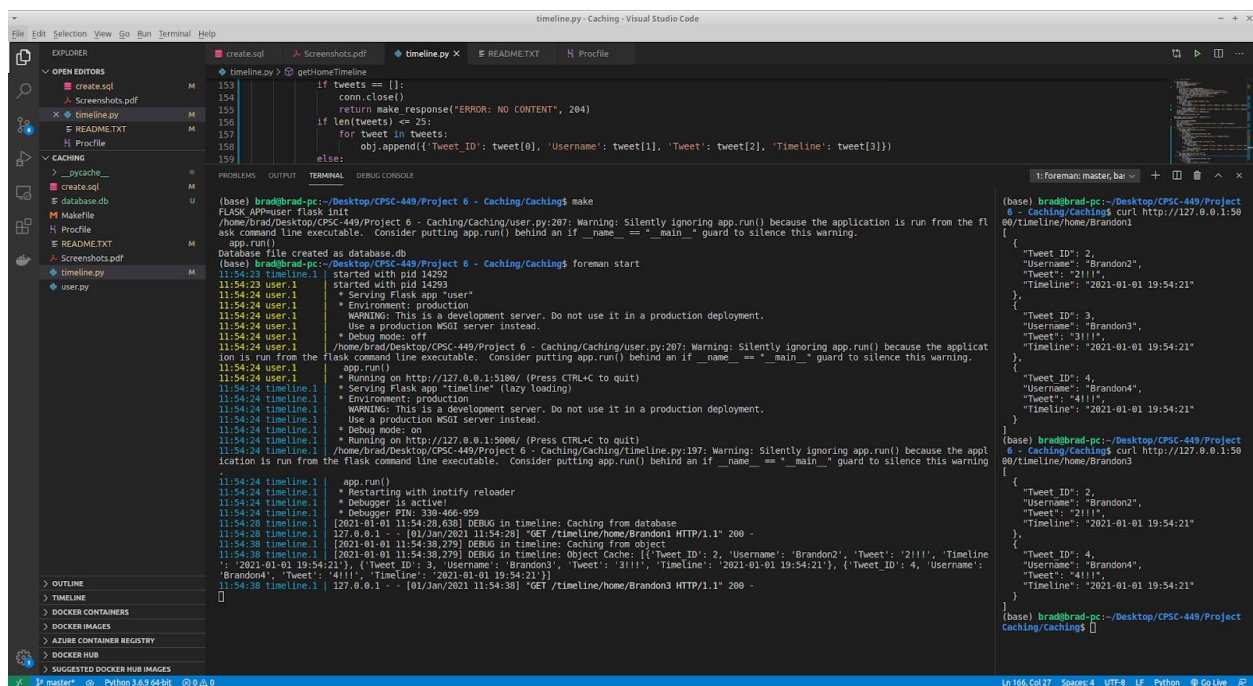
Called username Brandon1

```
$ curl http://127.0.0.1:5000/timeline/home/Brandon1
```

3. Calling `getHomeTimeline(username)` for a different user who follows some of the same users and verifying that the requests for those users' timelines are fulfilled from cache.

Called username Brandon3

```
curl http://127.0.0.1:5000/timeline/home/Brandon3
```



The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER:** Lists files including `create.sql`, `Screenshots.pdf`, `timeline.py`, `README.TXT`, `Profile`, `__pycache__`, `create.sql`, `database.db`, `Mahesh`, `Profile`, `README.TXT`, `Screenshots.pdf`, `timeline.py`, and `user.py`.
- EDITOR:** Displays the `timeline.py` file with the following code:

```
153 if tweets == []:  
154     conn.close()  
155     return make_response("ERROR: NO CONTENT", 204)  
156 if len(tweets) <= 25:  
157     for tweet in tweets:  
158         obj.append({'tweet_id': tweet[0], 'Username': tweet[1], 'Tweet': tweet[2], 'Timeline': tweet[3]})  
159 else:
```
- TERMINAL:** Shows the output of the application running. It includes messages about the database file, the Flask app starting, and the application running on `http://127.0.0.1:5000/`. It also shows the output of the `curl` command for `Brandon1` and `Brandon3`, which returns a JSON response containing tweet data.

```
(base) brad@brad-pc:~/Desktop/CPSC-449/Project 6 - Caching/Caching$ make  
FLASK_APP=user.py flask init  
Database file created as database.db  
11:54:23 timeline.1 started with pid 14292  
11:54:24 user.1 started with pid 14293  
11:54:24 user.1 * Serving Flask app "user"  
11:54:24 user.1 * Environment: production  
11:54:24 user.1 WARNING: This is a development server. Do not use it in a production deployment.  
11:54:24 user.1 Use a production WSGI server instead.  
11:54:24 user.1 * Debug mode: off  
11:54:24 user.1 /home/brad/Desktop/CPSC-449/Project 6 - Caching/Caching/user.py:207: Warning: Silently ignoring app.run() because the applicat  
ion is run from the flask command line executable. Consider putting app.run() behind an if __name__ == "__main__" guard to silence this warning.  
11:54:24 user.1 app.run()  
11:54:24 user.1 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)  
11:54:24 timeline.1 * Serving Flask app "timeline" (lazy loading)  
11:54:24 timeline.1 * Environment: production  
11:54:24 timeline.1 WARNING: This is a development server. Do not use it in a production deployment.  
11:54:24 timeline.1 Use a production WSGI server instead.  
11:54:24 timeline.1 * Debug mode: on  
11:54:24 timeline.1 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)  
11:54:24 timeline.1 /home/brad/Desktop/CPSC-449/Project 6 - Caching/Caching/timeline.py:197: Warning: Silently ignoring app.run() because the appl  
ication is run from the flask command line executable. Consider putting app.run() behind an if __name__ == "__main__" guard to silence this warning.  
11:54:24 timeline.1 app.run()  
11:54:24 timeline.1 * Restarting with InotifyReloader  
11:54:24 timeline.1 * Debugger is active!  
11:54:24 timeline.1 * Debugger PIN: 338-466-959  
11:54:28 timeline.1 [2021-01-01 11:54:28.638] DEBUG in timeline: Caching from database  
11:54:28 timeline.1 127.0.0.1 - - [01/Jan/2021 11:54:28] "GET /timeline/home/Brandon1 HTTP/1.1" 200 -  
11:54:38 timeline.1 [2021-01-01 11:54:38.279] DEBUG in timeline: Caching from object  
11:54:38 timeline.1 [2021-01-01 11:54:38.279] DEBUG in timeline: Object Cache: [{'tweet_id': 2, 'Username': 'Brandon2', 'Tweet': '2!!!!', 'Timeline'  
, '2021-01-01 19:54:21'], {'tweet_id': 3, 'Username': 'Brandon3', 'Tweet': '3!!!!', 'Timeline': '2021-01-01 19:54:21'}, {'tweet_id': 4, 'Username'  
, 'Brandon4', 'Tweet': '4!!!!', 'Timeline': '2021-01-01 19:54:21'}]  
11:54:38 timeline.1 127.0.0.1 - - [01/Jan/2021 11:54:38] "GET /timeline/home/Brandon3 HTTP/1.1" 200 -
```
- TERMINAL (Right):** Shows the output of the `curl` command for `Brandon1` and `Brandon3`, which returns a JSON response containing tweet data.

```
(base) brad@brad-pc:~/Desktop/CPSC-449/Project 6 - Caching/Caching$ curl http://127.0.0.1:5000/timeline/home/Brandon1  
{  
  "tweet_id": 2,  
  "Username": "Brandon2",  
  "Tweet": "2!!!!",  
  "Timeline": "2021-01-01 19:54:21"  
}  
{  
  "tweet_id": 3,  
  "Username": "Brandon3",  
  "Tweet": "3!!!!",  
  "Timeline": "2021-01-01 19:54:21"  
}  
{  
  "tweet_id": 4,  
  "Username": "Brandon4",  
  "Tweet": "4!!!!",  
  "Timeline": "2021-01-01 19:54:21"  
}  
(base) brad@brad-pc:~/Desktop/CPSC-449/Project 6 - Caching/Caching$ curl http://127.0.0.1:5000/timeline/home/Brandon3  
{  
  "tweet_id": 2,  
  "Username": "Brandon2",  
  "Tweet": "2!!!!",  
  "Timeline": "2021-01-01 19:54:21"  
}  
{  
  "tweet_id": 4,  
  "Username": "Brandon4",  
  "Tweet": "4!!!!",  
  "Timeline": "2021-01-01 19:54:21"  
}  
(base) brad@brad-pc:~/Desktop/CPSC-449/Project 6 - Caching/Caching$
```