MoodAPI Programming Challenge  
Tom McCaffrey, [tom@realhelp.net](mailto:tom@realhelp.net)  
215.740.0491

This API is currently implemented and running on the Digital Ocean server located at   
https:// 206.81.7.178, using a PostgreSQL database created specifically for this API.

This API contains the following four endpoints:

**GET** [**https://206.81.7.178:8000/**](https://206.81.7.178:8000/)- API health check that returns OK when the server is running.

**POST** [**https://206.81.7.178:8000/token**](https://206.81.7.178:8000/token)– Returns an OAuth 2.0/JWT secure web token that when securely unwrapped contains a username and token expiration time. The expiration time is currently set to 10 minutes. This POST endpoint takes as parameters the ‘username’ and ‘password’ of the desired user in a URL encoded form.

Note that for this example program, I have set up four users in the database: ‘tom’, ‘matt’, ‘ryan’, and ‘megan’ (all lowercase). The password is the same for all: ‘Neuroflow-1’.

**POST** [**https://206.81.7.178:8000/mood/{mood}**](https://206.81.7.178:8000/mood/%7bmood%7d) – This posts a mood to the database for the user defined in the token from the /token endpoint. Possible values for {mood} are ‘happy’, ‘sad’, ‘angry’, or ‘OK’. The header of the request must be set to Content-Type application/json with Authorization set to the word ‘Bearer ‘ followed by the token received from the /token endpoint. When successful, it returns a 200 status code and returns all of the mood records for the user.

**GET** [**https://206.81.7.178:8000/mood**](https://206.81.7.178:8000/mood) - This performs all of the requested functions for the GET method as described in the programming challenge document. The header of the request must be set to Content-Type application/json with Authorization set to the word ‘Bearer ‘ followed by the token received from the /token endpoint.   
  
When successful, it returns a 200 status code and returns the following:

* The length of the current streak for the user.
* The length of the longest streak for the user.
* The streak percentile of the user’s longest streak as compared to all other users (returned only if the user’s percentile is greater or equal to 50%).
* All mood records for that user.

**IMPORTANT NOTE: For this example, SSL encryption has not been enabled, which of course would be REQUIRED in a production environment. For this example, please use http:// for the requests instead of https:// . For production, we would need to deploy this to a server with a current SSL certificate.**

**Deployment**

This project was deployed to the Digital Ocean server by using the following docker commands:

docker build -t mood-api-build .

docker run -idt --name=mood-api -e MODULE\_NAME="main" -e PORT="8000" -p 8000:8000 mood-api-build

**Things that would need to be changed for production:**

The following would be issues I would recommend to be changed if this API were used in a production environment:

1. Deploy to a server with a current SSL certificate and a properly configured firewall. In particular, the server containing the PostgreSQL database should have the firewall configured to not accept any connections from any IP addresses except from the API server and a small list of developers who are actively supporting it.
2. The algorithm for calculating the current streak and the streak percentiles would need to change. Currently, because the streak percentile requires a determination of the maximum streak length for each user, it currently requires that the ENTIRE moods table be scanned. I have implemented this an ‘easy to read’ for loop that loops through the entire table ordered first by username and then by date, erring on the side of being easier to read vs. remarkably clever programming. I also am using an ‘epoch date’ format for the calculations, i.e. the number of days since January 1, 1970, which in addition to making the streaks easier to calculate because we’re working with an integer, also has the advantage that there is no need to convert the dates back and forth from standard date formats. Please see the comments in the db\_functions.py file for the exact implementation details.  
     
   The solution to this problem would be to add a database file ‘streaks’, which contained at minimum the following fields:

username: str,   
current\_streak\_cnt: int,

current\_streak\_epochday: int,

longest\_streak: int,

longest\_streak\_end\_epochday: int  
  
What you would do basically would be to check the current\_streak\_epochday vs. the current epochday; if the current is only one day greater, then you'd update the current\_streak\_cnt by one. If the current streak is now greater than the longest\_streak, you'd also update the longest\_streak +1 and longest\_streak\_end\_epochday to the current day.

1. Finally, some of the return values would probably be changed or removed for production; there’s really no need to provide an entire list of all moods for each user for the GET and POST mood endpoints.

Any questions, please feel free to email me at [tom@realhelp.net](mailto:tom@realhelp.net). Thanks!