2nd Lab Project - Fitbit API, MongoDB & Streamlit

Web Data Mining - SS 2023

This is a hands-on lab project on using public APIs for acquiring data from the web, storing them in NoSQL databases, and visualizing them on a web platform. Each group (same as 1st lab project) will experiment on a task that contains Fitbit data (acquired from the Fitbit Web API), Python, MongoDB and Streamlit. **The group will have to prepare a consistent, detailed hands-on tutorial as a Medium[1] article (maximum 8 pages)**, focused on the general topic of "Using the Fitbit Web API with Python, MongoDB and Streamlit"; no further report will be required. An indicative example of a Medium tutorial article can be found here or here.

The tutorial will have the following sections:

A. Creating a Fitbit developer account and registering an application
B. API authorization in Python
C. API requests for data acquisition
D. Data storage in MongoDB
E. Setting up a Streamlit website with Python
F. Visualizations with Streamlit for understanding Fitbit data
G. Insightful conclusions—what you've learned

Don't forget to:

- Document the process and **your conclusions** in your Medium-like article
- Share your code with Jupyter Notebook or Github repository
- Upload/send your solutions until **08/05/2023**

Topics

Fitbit is a consumer electronics and fitness company. It produces wearable technology, physical fitness monitors, and activity trackers such as smartwatches, pedometers and monitors for heart rate, quality of sleep, and steps as well as related software. For this project, **you are requested to acquire data from the Fitbit Web API for distinct fitness indicators**. Specifically:

Table 1: Team allocations

Team	Topic
[112 - 92 -122 - 104]	Sleep & Breathing Rate

[1] https://medium.com

[Team #2]	Steps & Activity
[126 - 150 - X]	Heart rate & Activity
[140 - 138 - 151]	Sleep & Heart Rate Variability
[142 - 135 - 139]	Sleep & User Engagement
[148 - 147 - 62]	A day in a user's life
[Team #7]	Steps & User Engagement

Attention

- To get intraday data (i.e., finer granularity data corresponding to second, minute, or hour measurements) use the Intraday endpoint. Take care, to gain access to this endpoint, make sure you register a "personal" application. Instructions on how to create a developer profile and register and application can be found here.
- There is a **minimum requirement of 7 days of data** for the purpose of the project.

Data Collection

We will provide at least one team member with a Fitbit Sense for the full duration of the lab project in order to enable data collection. Please e-mail the teaching assistant Christina Karagianni (kechristi@csd.auth.gr), who can help you with the device if necessary.

For collecting data, you do not need to reinvent the wheel. To set up your account, register an application and acquire your Fitbit data (Steps A, B, C), you can follow the steps mentioned in this tutorial (this is just a hint and by no means the only way to fulfill the requirements). It uses the unofficial Python Fitbit API wrapper, also found on GitHub. Alternatively, you can use cURL requests in Python, as seen in the official Fitbit documentation (e.g., cURL requests and responses for intraday activity data).

Figure 1: An example response from the Fitbit Web API

The Fitbit Web API is well-documented, and all its endpoints can be found here. Each team needs to acquire data from the endpoints relevant to their topic, e.g., the ”Steps & Activity” team can acquire data from the “Active Zone Minutes”, “Activity”, “Activity Time Series”, and relevant “Intraday” endpoints, or, optionally, correlated measures, such as “Cardio Fitness Score”. An example response can be seen in Figure 1.

Data Storage

Once you acquire your data with Python, you should store it in a (local) MongoDB. MongoDB is a cross-platform document-store database. Classified as a NoSQL database, MongoDB uses JSON-like (BSON) documents with optional schemas. For usability purposes, we recommend you use MongoDB Compass or Studio 3T GUIs for visualizing your database contents.

Each document in your MongoDB should follow the format of Figure 2.

- **_id:** This is a MongoDB automatically generated field and it represents the unique document ID.
- **id:** This is the user ID. You can use a random SHA2 Hash Generator to create an ID for each user.
- **type:** Since you will be working with multiple data types (e.g., activity, steps, etc.), you need a way of distinguishing them in the database. You can use the key “type” for this purpose.
- **data:** This is the actual data returned from the Fitbit API. Take care that each document in the collection contains **only one Fitbit record**. For example, the Fitbit response might include nested JSONs with multiple step values per document. You should split it as shown in Figure 2.

Figure 1: An example response from the Fitbit Web API

Data Visualization

In this step, you will create and share some interesting visualizations to get a better understanding of your data. For a faster way to build and share your data app, we will use Streamlit, an open-source Python library that makes it easy to build beautiful custom web apps for machine learning and data science. You do not need any front-end or JavaScript knowledge to use it. With Streamlit, you should be able to read data from MongoDB and visualize them in a custom web platform.

Feel free to get creative with your visualizations, but at least include user statistics and ten distinct **interactive** visualizations related to your topic. You need to share the URL of your web platform, including all deliverables (medium-like article, Github repo, or Jupyter notebook, and statistics and visualizations) with the teaching assistants.

Bonus Questions

If you enjoy the project, you can advance your knowledge by completing any of the following bonus questions. Each bonus question with **add 10% to your project grade** (not your final grade overall). Specifically:

- Instead of reading data offline, utilize the Fitbit Subscription endpoints to create a "listener" waiting for **live user updates**. Fitbit will send a webhook notification informing the application that the user has new data to download. This way, your application will continuously receive new user data upon availability on the Fitbit server. Alternatively (5% bonus) you can achieve a similar functionality through a cron job setup.
- Beyond visualizations, how could you integrate **machine learning** into your application's pipeline? Develop and deploy a machine learning algorithm for a task relevant to your topic.