

Grafana Web App Backend

Elton Lam, Feb 27, 2023

Overview

Our web application visualizes patient data obtained from Fitbits, Withings Sleep Tracking Mats, and Polar Chest Straps. It consists of a locally hosted web application with features to add patients and connect them to their device API. The web application can also run several scripts to directly download data from device api to local mysql databases, which are connected to a web-based data visualization application called grafana. In particular, the grafana instance is a set of customizable dashboards made up of widgets that directly query and visualize the mysql data. In addition, we are exposing our local grafana server and web application to the internet by using ngrok, a tool that creates secure tunnels using a reverse proxy.

Set Up:

To host the web application, grafana application, and mysql database, follow the instructions below.

Installing MySQL:

Follow the instructions for installing MySQL:

[MySQL 8.0 Reference Manual :: 2.4 Installing MySQL on macOS](#)

For the mac instructions:

Enter `brew install mysql` followed by `brew services start mysql` to manually start the mysql server. **Note** that you have to run the latter command every time you restart the system.

[2.3 Installing MySQL on Microsoft Windows](#)

From the windows instructions:

The simplest and recommended method is to download MySQL Installer (for Windows) and let it install and configure a specific version of MySQL Server as follows:

1. Download MySQL Installer from <https://dev.mysql.com/downloads/installer/> and execute it.

Note

Unlike the standard MySQL Installer, the smaller `web-community` version does not

bundle any MySQL applications, but downloads only the MySQL products you choose to install.

2. Determine the setup type to use for the initial installation of MySQL products. For example:

- **Developer Default:** Provides a setup type that includes the selected version of MySQL Server and other MySQL tools related to MySQL development, such as MySQL Workbench. (Note: as of Feb 27, 2023, this setup type requires visual studio installed on the computer, if you plan to use a different IDE with mysql, you can do a custom install instead and uncheck the "MySQL for Visual Studio" option)
- **Server Only:** Provides a setup for the selected version of MySQL Server without other products.
- **Custom:** Enables you to select any version of MySQL Server and other MySQL products.

3. Install the server instance (and products) and then begin the server configuration by following the onscreen instructions. For more information about each individual step, see [Section 2.3.3.3.1, "MySQL Server Configuration with MySQL Installer"](#).

MySQL is now installed. If you configured MySQL as a service, then **Windows automatically starts the MySQL server every time you restart the system**. Also, this process installs the MySQL Installer application on the local host (:3306), which you can use later to upgrade or reconfigure MySQL server.

When prompted to select the setup type, select 'Developer Default'.

Creating MySQL Users

For the python scripts and the grafana instance to modify and query the database, we need two users, one capable of writing data and one capable of reading.

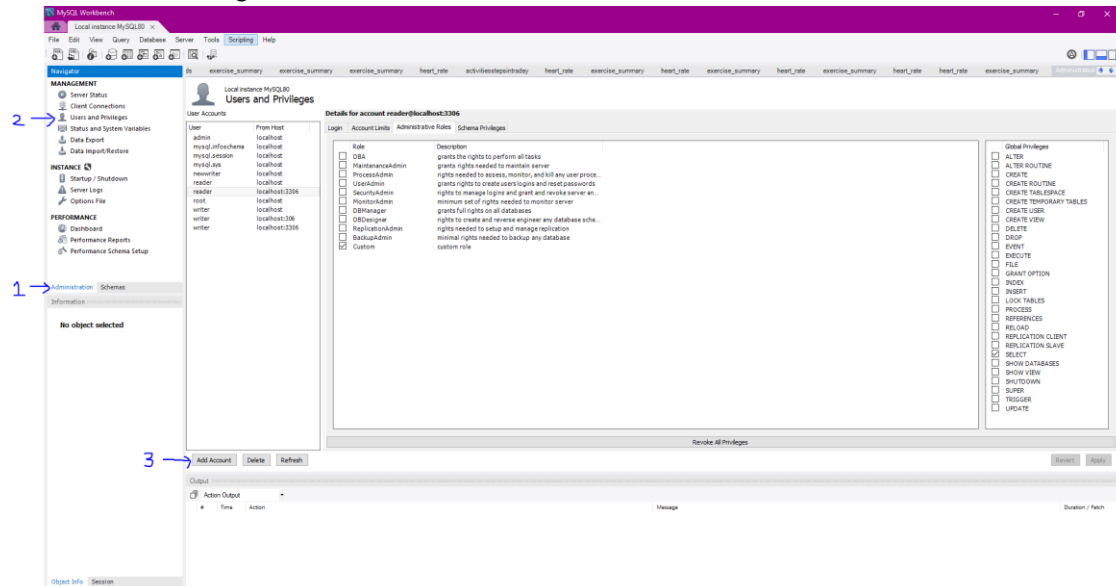
The following commands create two such users:

```
mysql> CREATE USER 'writer'@'localhost' IDENTIFIED BY 'password';  
mysql> GRANT ALL PRIVILEGES ON *.* To 'writer'@'localhost';
```

```
mysql> CREATE USER 'reader'@'localhost' IDENTIFIED BY 'password';  
mysql> GRANT SELECT ON *.* To 'reader'@'localhost';
```

These two users are both identified by the password 'password'. Since 'writer' is being used by the python scripts to edit the databases, it has been granted all privileges (i.e. the ability to run any kind of query on any database). In contrast, 'reader' is only being used by the grafana instance, so it only needs to be able to run 'SELECT' queries.

Alternatively, open the MySQL Workbench and navigate to: Administration -> Management -> Users and Privileges -> Add Account.



Set the 'Authentication Type' to 'caching_sha2_password' and 'Limit to Hosts Matching' to 'localhost'. Click on Administrative Roles and check all Global Privileges and click apply. This user will be the **writer**.

Next, add another user repeating the steps above but only check 'Select' for Global Privileges. This user will be the **reader**.

Setting Up MySQL Databases: Windows and Mac:

[MySQL 8.0 Reference Manual :: 3.3.1 Creating and Selecting a Database](#)

After completing the installation, enter the `mysql -uroot` command to open the MySQL monitor as the root user. By default, the server can be accessed at localhost:3306. To display the port enter the following query in the monitor: `mysql> SHOW GLOBAL VARIABLES LIKE 'PORT';`

Firstly, we need to set up all the databases to store all our tables. Once the MySQL Users are set up, write the username and password of the MySQL writer user into environment variables labeled as 'SECRET_USER' and 'SECRET_PASSWORD'. For more information on setting up environment variables, please refer to the "Setting up Device API: Setting up environment variables" section below.

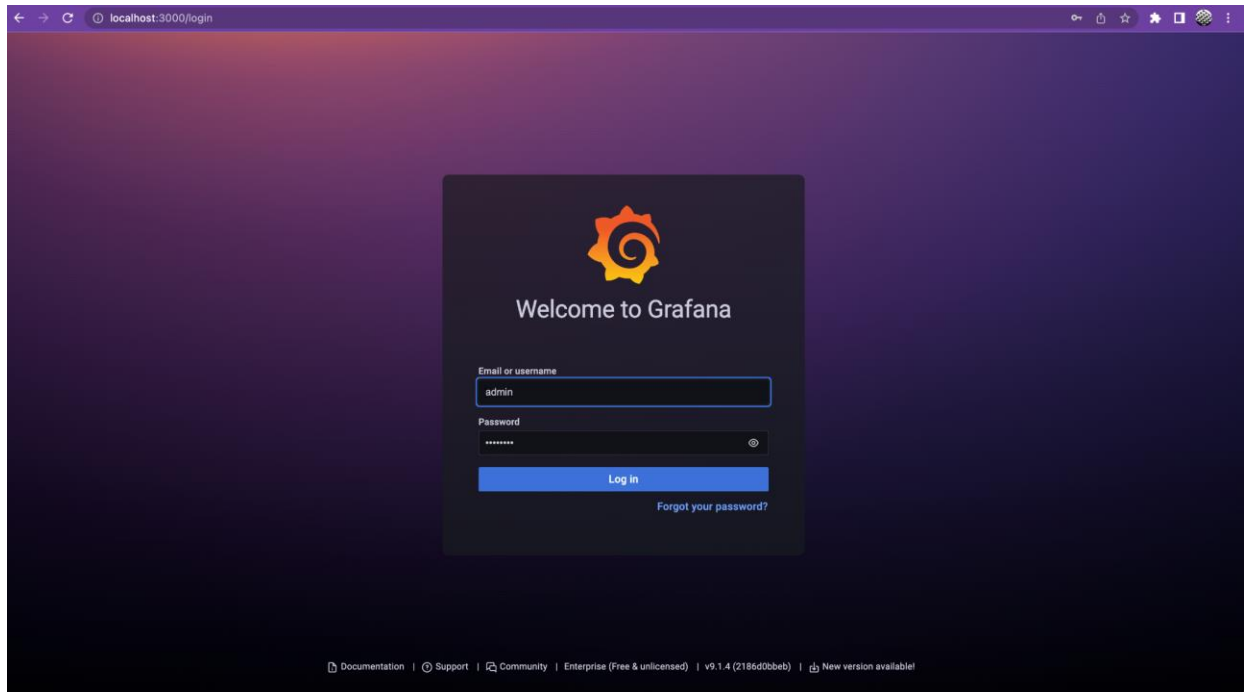
Next, please make sure to pip install all the python libraries in the requirements.txt file and then run the setup.py file to create all the databases.

If you need help installing the libraries, please see this [link](#):

Installing & Setting up Grafana Instance (Windows & Mac):

Follow these instructions [Download Grafana | Grafana Labs](#) to download grafana.

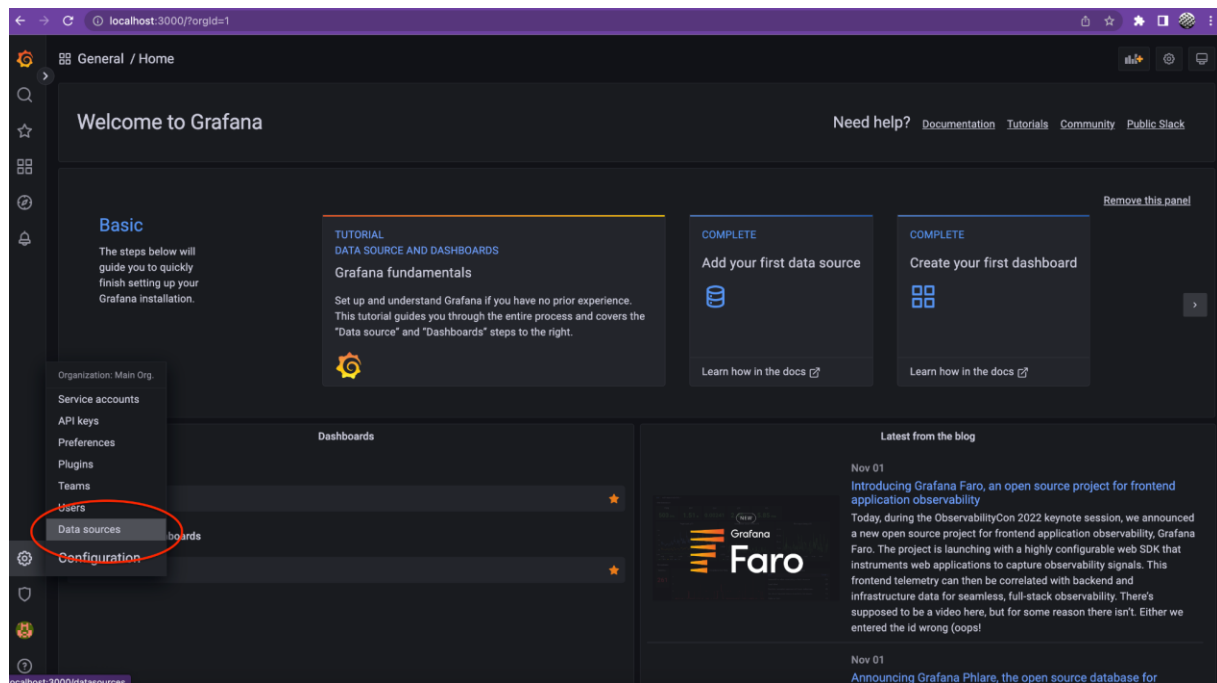
Once you've downloaded grafana, start the web server by entering the `./bin/grafana-server` while in the grafana directory if you're using a mac. On windows the web server should be running once you've completed the installation. Try opening localhost:3000. It should look something like this:



The default admin (editor) login credentials are 'admin' for both the username and password.

Connecting the Grafana instance to the MySQL database:

After logging in as the admin go into the 'Data sources' settings.



Then search for the MySQL datasource. Once you've selected it, enter the following parameters:

The screenshot shows the Grafana interface for configuring a MySQL data source. At the top, the MySQL logo is visible next to the title 'Data Sources / withings' and the text 'Type: MySQL'. Below this is a 'Settings' tab. A green status bar indicates 'Alerting supported'. The 'Name' field is set to 'fitbit', and a 'Default' toggle switch is turned on. The 'MySQL Connection' section contains fields for 'Host' (localhost:3306), 'Database' (fitbit), 'User' (writer), and 'Password' (configured). A 'Reset' button is located next to the password field. Below these are 'Session timezone' (set to default), 'TLS Client Auth' (disabled), 'With CA Cert' (disabled), and 'Skip TLS Verify' (disabled). The 'Connection limits' section has 'Max open' (unlimited), 'Max idle' (2), and 'Max lifetime' (14400). The 'MySQL details' section has a 'Min time interval' (1m).

The above picture is an example of setting up the Fitbit database to connect with Grafana. Please repeat the above step for the Withings and Polar databases as well.

Clicking on the 'Save & Test' button on the bottom left should result in a 'Database Connection Ok' message at the bottom.

Now all that's left is to create a dashboard full of panels that query the mysql database.

See this to learn how to use and build dashboards [Dashboards | Grafana documentation](#).

Setting up Web Application

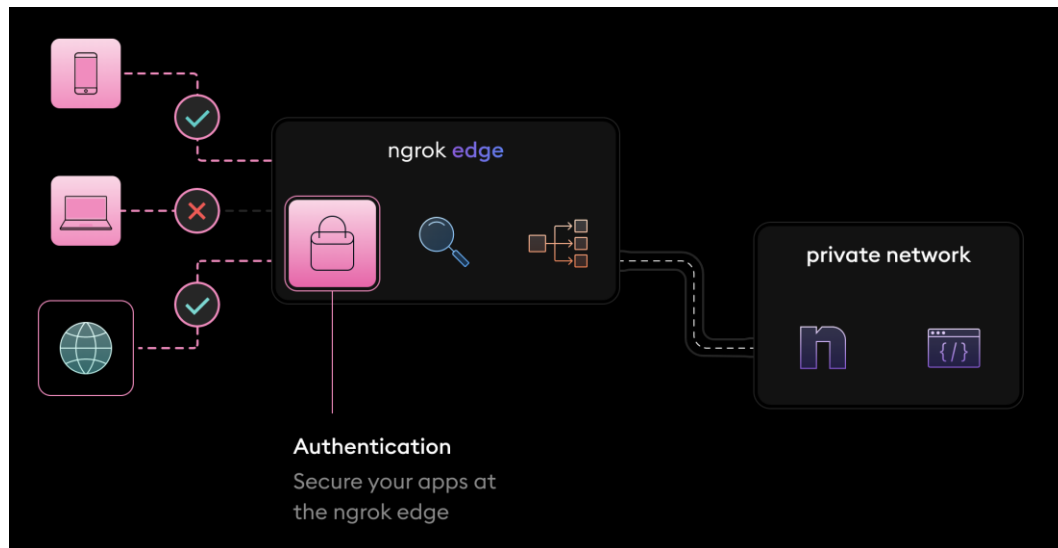
The web application is running on localhost:5000. This port was chosen at random and can be changed, however this change must be specified to the ngrok tunnel. To start the web application, run the run_web_app.py file.

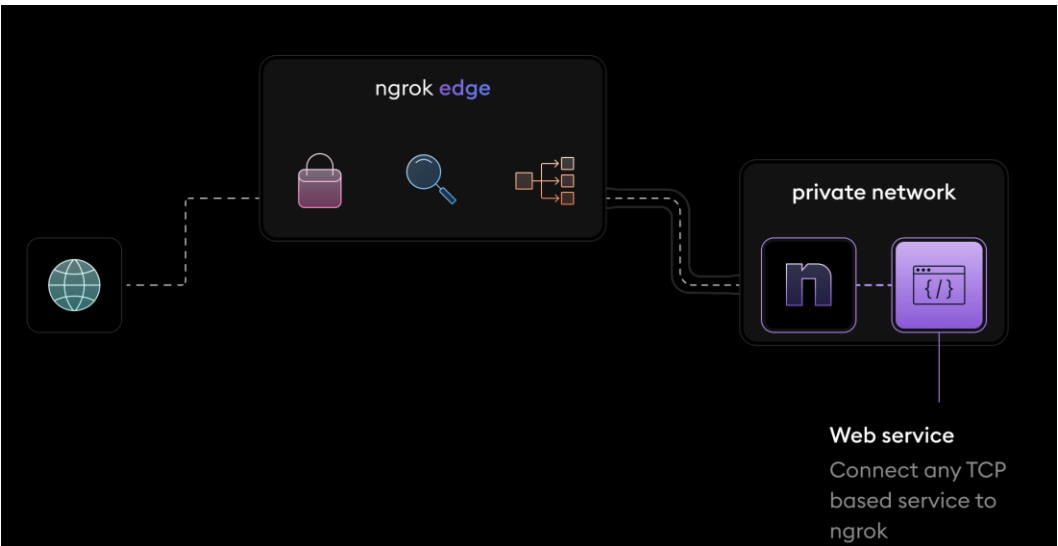
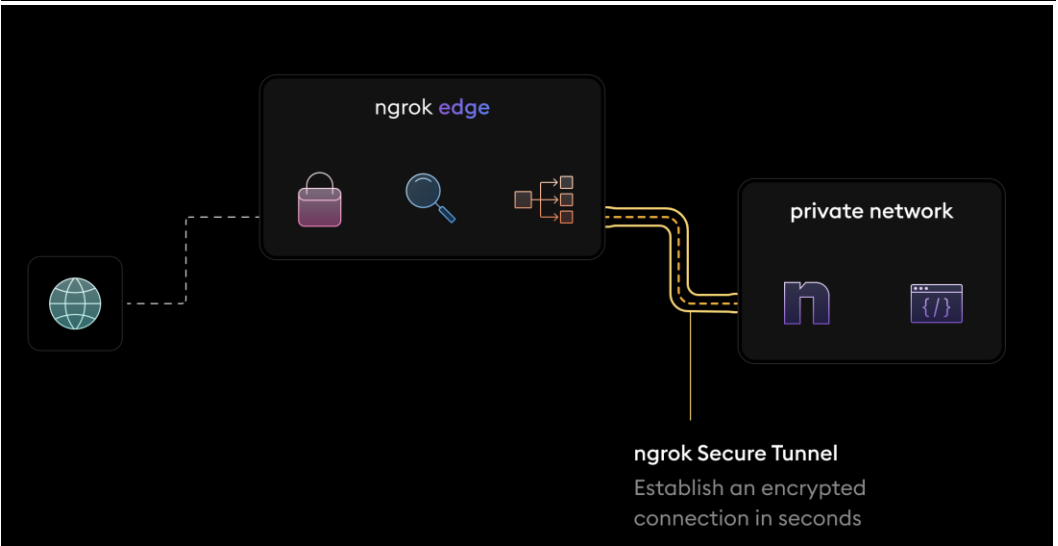
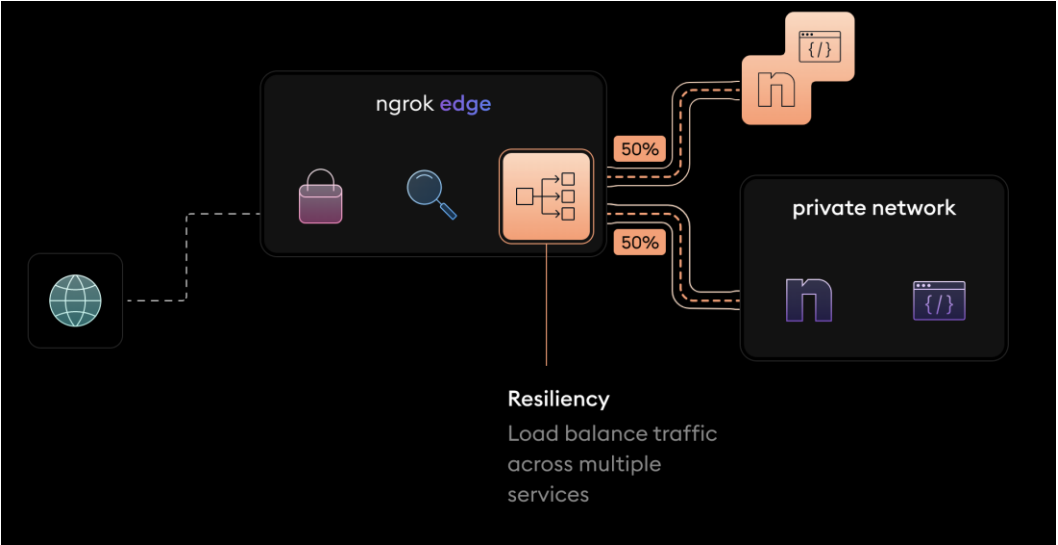
Setting up the ngrok Tunnel:

By default the grafana instance and web application will only be running on localhost:3000 and localhost:5000 respectively. The fastest way to expose our local server to the internet is to use a tool like [ngrok - download](#). “ngrok is a globally distributed reverse proxy fronting your web services running in any cloud or private network, or your machine.” - [ngrok](#).

Grafana comes with its own login system before being able to see patient data. This adds a second layer of security on top of the web application and can be used to separate user privileges. Those who have access to the web application can control adding and deleting patients while those with Grafana access can only view patient data.

I found the following graphics (from the ngrok site) to be helpful in understanding how it works.





To start using it you first need to make an ngrok account to get an auth token and add the auth token to your ngrok config file. See the installation and setup instructions on how.

Note: If you're using the free tier of ngrok you can only have one tunnel running at a time. So if you want to create another tunnel then you have to either take down your older tunnel or create a new account for a new authtoken.

However, there is a workaround. Look for the ngrok.yml file in your device. For windows, this can be found at C:\Users\{YourUsername}\AppData\Local\ngrok. If you cannot find this file, try first running

`ngrok config add-authtoken <token>` command in the ngrok terminal.

Open the file with notepad and replace all with the following:

```
ngrok - Notepad
File Edit Format View Help
version: "2"
authtoken:
tunnels:
  first:
    addr: 3000
    proto: http
  second:
    addr: 5000
    proto: http
```

Make sure to enter your authtoken into the correct location.

Now run the ngrok.exe application and enter the following command:

`ngrok start --all`

After running the above command, the bash terminal should look something like this

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
ngrok (Ctrl+C to quit)

Visit http://localhost:4040/ to inspect, replay, and modify your requests

Session Status      online
Account             Ragur (Plan: Free)
Version             3.1.0
Region              United States (us)
Latency              54ms
Web Interface        http://127.0.0.1:4040
Forwarding           https://d44f-142-244-4-194.ngrok.io -> http://localhost:3000

Connections          ttl    opn    rt1    rt5    p50    p90
276                  3      0.00   0.00   11.72  488.59

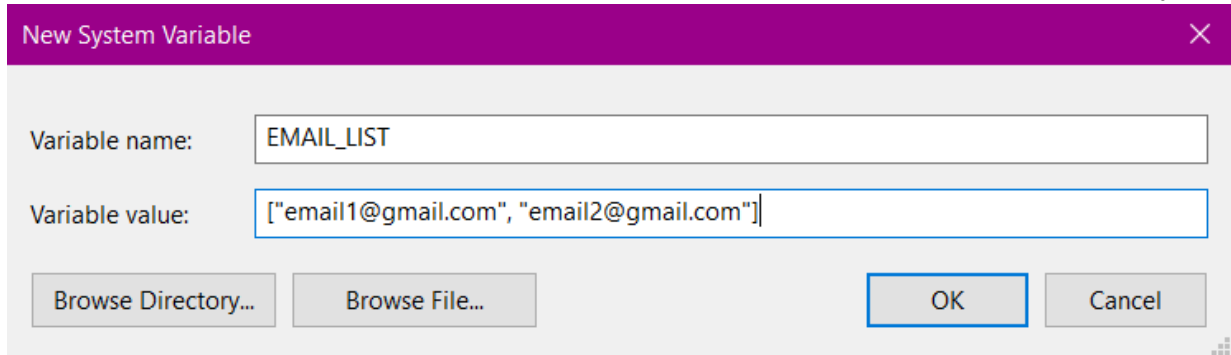
HTTP Requests
-----
GET /api/annotations                200 OK
POST /api/ds/query                  200 OK
POST /api/ds/query                  200 OK
POST /api/ds/query                  200 OK
POST /api/ds/query                  200 OK
POST /api/frontend-metrics          200 OK
GET /public/fonts/roboto/RxZJdnzeo3R5zSexge8UUVtXRa8TVwTICgirnJhmVJw.woff2 304 Not Modified
GET /public/fonts/roboto/CWB0XYA8bzo0kSThX0UTuA.woff2 304 Not Modified
POST /api/ds/query                  200 OK
GET /public/img/grab_dark.svg       304 Not Modified
GET /avatar/46d229b033af06a191ff2267bca9ae56 200 OK
```

For more info on running multiple tunnels on the free plan, see this [link](#).

In case of any changes to the ngrok urls, all the latest urls are updated in this google doc: [Links](#).

Setting up Email Notifications and Updater:

First create an environment variable named “EMAIL_LIST” and add emails following the data structure shown below. These emails will be notified when a user’s fitbit device is out of sync.



The screenshot shows a 'New System Variable' dialog box. It has a title bar with a close button. Inside, there are two text input fields. The first is labeled 'Variable name:' and contains the text 'EMAIL_LIST'. The second is labeled 'Variable value:' and contains the text '["email1@gmail.com", "email2@gmail.com"]'. Below these fields are four buttons: 'Browse Directory...', 'Browse File...', 'OK', and 'Cancel'.

Next add an email sender with variable name “DATA_PLATFORM_EMAIL” and password with variable name “DATA_PLATFORM_PASSWORD” to the environment variables.

The email must be gmail. For the password, please use a password generated from <https://myaccount.google.com/apppasswords>.

The application is capable of retrieving user data every day. To start the updater, run the daily_update.py file. This program should run 24/7 and will update all user data at 12:00AM. Afterwards, it will check which users haven’t synced their **Fitbit** device in the last 3 days and send an email to those listed in the environment variables.

Setting up Device API:

For the platform to read user data, we must first develop an application and then add users to the application.

To develop an application, first navigate to the device API website and create an application.

[Fitbit](#)

[Withings](#)

[Polar](#)

Note: Applications have already been developed under pathedmonton@gmail.com

Setting up redirect URL:

When creating an application, you should see an option to enter a redirect url. This url is important for navigating back to the web application after the API call. For the redirect url, you need to enter the web application ngrok tunnel url followed by “/oauth2_callback”.

So for example if your ngrok tunnel was:

`https://5cc0-142-244-4-194.ngrok.io -> http://localhost:5000`

Then the redirect url should be:

https://5cc0-142-244-4-194.ngrok.io/oauth2_callback

Once created, you should get a CLIENT ID and a CLIENT SECRET for the application. For security purposes, we store this information locally on the host computer as an environment variable.

Setting up environment variables:

For mac, follow these instructions:

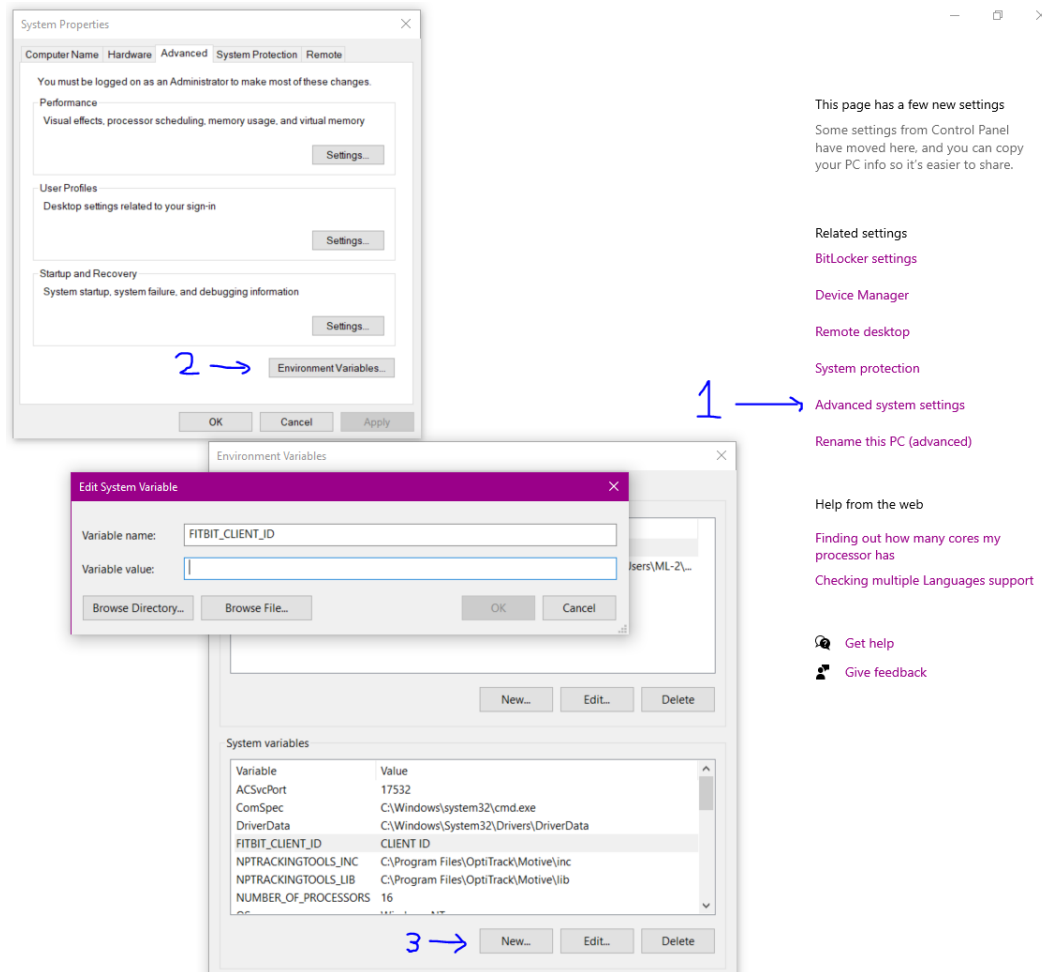
[How to Set Environment Variables in MacOS](#)

On Windows 11, right click on the start button in the bottom left corner and click system. Next navigate to Advanced system settings -> Environment Variable. Here you enter your Variable name and Variable Value.

For Fitbit, please enter the variable names as "FITBIT_CLIENT_ID" and "FITBIT_CLIENT_SECRET".

For Withings, please enter the variable names as "WITHINGS_CLIENT_ID" and "WITHINGS_CLIENT_SECRET".

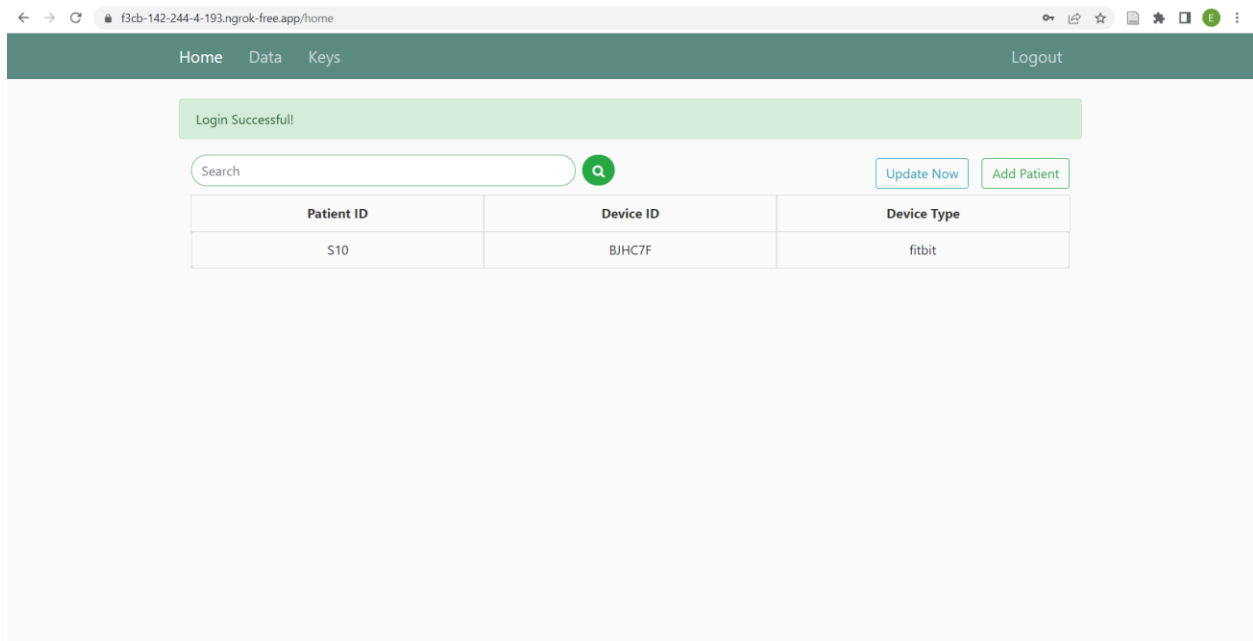
For Polar, please enter the variable names as "POLAR_CLIENT_ID" and "POLAR_CLIENT_SECRET".



For details on how to get data from the device API, please refer to the manufacturer documentation. [Fitbit](#), [Withings](#), [Polar](#).

If you are using ngrok, the web application and grafana urls must also be stored in an environment variable called "DATAPLATFORM_URL" and "GRAFANA_URL".

Navigating Web App:



Creating Web App Users:

To register an account for the web application, the user requires a registration key. During the initial setup, there is an automatically generated key with value "0123456789". To add new keys,

Go to the 'Keys' tab on the web app navigation bar. From there click generate and a key should appear for others to use. Unused keys are removed at reset so please use the key before then.

Adding Patients:

Once logged in, patients can be added by clicking the Add Patient button. Each patient ID must be unique across similar devices, but can be reused across different devices (ex. You can have patient_1 for a Fitbit and Withings device simultaneously, but you can't have patient_1 for two Fitbit devices simultaneously). When adding a patient, the user will be redirected to the manufacturer's login portal and must login to successfully add a new patient.

Updating Data:

To update data, click on the update now button. This will fetch data for any patients who haven't already been updated that day. For Polar devices, this will only fetch data if a new activity has been created.

Editing Patient:

On the home page, click on a patient to access their profile. The user can then choose to delete the patient. Note: this will also delete all data related to this patient in the database.

The user may also choose to change the patient ID of the patient. Clicking “change” will only change the patient ID corresponding to the selected device type. Clicking “change all” will change the patient ID across multiple devices.

Adding New Technology:

Every API is different so you will have to look for documentation on how the manufacturer has set up their device API.

General steps though include:

1. Creating a database for the new device and structure it to store the data that will be recorded.
2. Adding the device as an option in the web application.
3. Connecting the new database to Grafana and adjusting how it should look.
4. Figuring out how to store auth keys or tokens in the database.
5. Writing a method to update the data automatically, manually, or both.

The current platform hosts 3 different devices already: Fitbit, Withings, and Polar. All of these devices use OAuth authentication. See their files for an example of how they have been implemented.

Step By Step Guide To Start Web App:

For cases when the server computer shuts down and closes all the applications.

1. Start ngrok, grab and copy the tunnels to the localhost ports 3000 and 5000.
2. Change web links in system environment variables and google doc
3. Change callback links in Fitbit, Withings, and Polar API
4. Start the web application `web_app.py` and start the daily updater `daily_update.py`

Design Reasoning:

Why did we use Grafana to visualize the data?

The Grafana application served as a quick and simple way to display data and connect to our MySQL database. It had its own login for security and was easy to expose to the internet using the ngrok tunnel. In the future, it may be best to display data directly from the web application.

Why did we need a web application?

The web application gives the user a way of remotely adding and removing patients that are using Fitbit, Withings, or Polar devices.

Why did we use a SQL database?

Grafana's free tier doesn't have the option to connect to a NoSQL database, therefore the next best option was MySQL. A NoSQL database may be the better option in the future.

Why Python?

Python was the ideal choice for rapid application development.

Why Fitbit, Withings, and Polar?

These brands were chosen based on popularity, accuracy, and feasibility.

What do the devices measure and why?

The Fitbit watch measures steps taken, heart rate, and sleep. This allows clinicians to get data throughout the day on patients. The withings sleeping Mat measures sleep and does it much more accurately than the Fitbit watch does. The Polar H10 measures heart rate very accurately and is best used during high intensity activities (not practical or comfortable to use during the whole day due to software and design limitations).

Suggested Improvements:

- Verify how withings and polar data is saved to CSVs.
- Withings devices may miss data if the user forgets to synchronize the device during that day. It may be useful to verify if this occurs and fix it. (Could maybe use a similar method to how the Fitbit updater catches missed days?)
- Add functionality to web app to add and remove emails for email notifications
- Create an undo button in case a user changes or deletes a patient by accident
- Create two levels of users: admin and viewer
- Add a way of deleting certain datasets
- Add reset password to web app
- Check password complexity
- Add login timeout on unsuccessful attempts
- Add a way for users to fetch data for a certain time range. Currently, clicking update now on the web app will only get data from the last day
- Check what happens when a patient switches their fitbit for a new one
- Add functionality to the web app that lets users add data from a specified time range (Note: this function already exists in manual_update.py but it has not been added to the web app).
- Fix how the platform shuts down if the computer automatically turns off or logs out