

Appointment Scheduler

A part of our mission is to connect our members with excellent practitioners to help them access the quality healthcare they need and deserve.

We want to build a simple service to help users schedule appointments. To keep it simple, we offer coverage 24/7/365 (including weekends and holidays) and have unlimited practitioners (in other words, any number of users can schedule appointments for the same time).

Assignment

Your task is to build a **HTTP** service that serves two **JSON** endpoints:

- A **POST** endpoint that takes a date/time and user ID (both required). Creates an appointment beginning at that time for that user **or** returns an appropriate status code and error if the appointment cannot be made (the user already has an appointment that day).
- A **GET** endpoint that takes a user ID (required) and returns all appointments for the user.

The service has these requirements:

- all appointments must start and end on the hour or half-hour
- all appointments are exactly 30 minutes long
- a user can only have 1 appointment on a calendar date

Details

- Appointment date/time
- If you are familiar with a date library, such as Python's **datetime** or Node/Express's **moment** or **date-fns**, feel free to use that. You can assume all timestamps are in UTC (no timezone handling required).
- If you are less familiar with these libraries, no problem! You can assume the date and time are given as delimited strings, and are all in the same time zone.

Example

```
{
  "date": "2020-11-04", // calendar date in YYYY-MM-DD format
  "time": "09:00" // HH:MM format, with 00:00 being midnight and 13:30 being 1:30 PM
}
```

- You may represent the date and time in a single value, or in separate fields. Whichever you choose, make sure the API usage is clearly documented.
- If this would impact your design, you can assume input dates will be "recent" – e.g. today, within the past few days or next couple of weeks. In other words, you don't need to worry about dates being extremely far in the past or future.
- User ID this can be any reasonable format, such as integer, email (string), or UUID. You do not need to handle user creation, authentication, or authorization.

- Your server does not need to store data across restarts. It is fine to use an in- memory store.
- Feel free to add any extra validation or error handling if you have time.
- If you find any ambiguities, open questions, or tradeoffs, please make any reasonable decision and briefly discuss your reasoning in your README. If you have questions or anything you'd like to clarify, feel free to reach out to your recruiter!

Language

Use Python as a backend language and feel free to look up documentation and use standard libraries and common open- source libraries like Flask or Django.

Submission

Please submit a ZIP/tarball containing all files or a Github/Gitlab/Bitbucket repository link.

In addition to your source code, README, and dependencies files (like requirements.txt or package.json), you may include other files like helper scripts or unit tests. However, these additional files are NOT required.

README

Documentation is an important part of our engineering culture, so we'd like to see how you document a project so that others can use it.

Your submission must include a README file that contains instructions on how to run your service and make requests to the API.

Options:

1. Make a docker image publicly available on Docker Hub, so that someone can take a computer with docker installed on it and run a command in the README like:

```
docker run --rm -p 80:80 myrepo/myimage
```

2. Include a Dockerfile to build the image locally. Please include instructions for how to build and run the image.
3. (Non-Docker) Include commands on how to build and run your code locally, including any dependencies that need to be installed first and any specific versions required. Our engineers typically use Mac OS X and Linux, so please note if your code must be run on Windows.

Note: using (or not using) Docker does not affect our evaluation. The option is offered to make it easier to ensure we can run your code in the exact environment you intended.

Time

This exercise is intended to take less than 2 hours. Please spend no more than 2.5 hours total, including finalizing and sending your submission.

If you don't get around to everything you'd like to do, describe in your README or add TODO comments to indicate what you'd like to add/change given more time