Gordian Software Technical Take-Home

Gordian Software Technical Take-Home **1**
Overview 1
Tasks 1
Task Materials 2
Task 1: Direct interaction with our API 3
Task Overview 3
Steps 3
Task 2: Parse seatmap 3
Overview 4
Steps 4
What's next? 4

Overview

Welcome to Gordian! We appreciate you taking the time to do this test & apply to be part of our team. This take-home test is broken down into 2 tasks:

1. **Task 1**: A direct interaction with our API
2. **Task 2**: Create a seatmap parser

Important:

- **Although you will have unlimited time on technical take-home, we strongly recommend you to cap your time to 3 hours.** This test is not meant to take up an absurd amount of your time, but rather it is meant to address the following parts of our work in the content team: strong product understanding, solid code structure, and coding communication.
- The use of lxml is ***not allowed*** for this test. However, you are ***allowed to use Python's own library elementTree*** for parsing through the XML files.
- Write down any comments/questions/notes you have about any portion of the test in `comments.txt`.
- Include any libraries you used in `requirements.txt`.

Suggestion:

- We suggest you sit down in front of your computer without any distractions, and set an alarm for 2 hours. If you are unable to complete task 1 at the 2 hour mark, move to Task 2 to create the seat parser.

Tasks

Task Materials

You should be given the following. Please let us know if you have not received these:

1. An api key
2. A link to Gordian's documentation
3. A folder with the following structure

```
Unset
Folder name: [your_name]_gordian_content_interview
├── task_1
|    ├── input
|    |    ├── trip_1.json
|    |    ├── trip_2.json
|    |    ├── trip_3.json
|    |    ├── trip_4.json
|    ├── output
|    |    ├── trip_1.json
|    |    ├── trip_2.json
|    |    ├── trip_3.json
|    |    ├── trip_4.json
|    ├── script.py
|    ├── trip_3_seatmap.png
├── task_2
|    ├── seat_parser.py
|    ├── seatmap.xml
|    ├── seatmap.json
|    ├── example_output.json
|    comments.txt
|    requirements.txt
└── readme.md
```

Task 1: Direct interaction with our API

Task Overview

Given 4 itineraries in the input folder, create a **script** that search for the seatmap of that flight itinerary (a json file containing information about seats in a flight—can be pricing, availability, seat categories, etc) on Gordian's platform, and **save the outputted seatmaps** into an output folder.

Steps

1. Understand the files & navigate the website
    a. Navigate to `/task_1`, and open up the `/input` folder. You should see 4 json files in total. They indicate 4 different itineraries.
    b. Open up one of the trips. You will see a search object (in this case, we only care about seats), a ticket object that contains the exact segment to search for, and a passengers object containing passengers to search for. Examine different portions of this file.
    c. Go on the Spirit airlines website and search for trip 3's itinerary. Navigate to their seat selection page, and screenshot the seatmap.
    d. Save your screenshot in the folder as `trip_3_seatmap.png.`Note that this initial file is empty, and you should upload your picture with this file name. ***This only needs to be done for trip 3***.
2. Navigate Gordian's documentation
    a. Navigate to our documentation. Here are some helpful resources for understanding our basic concepts & flows:
        i. API basics
        ii. Offering ancillaries through Gordian
    b. Look into how we do the following with an itinerary. Your itinerary will come in handy for the create trip & search portion.
        i. Authorize for the user (authorize with your API key, and obtain a token)
        ii. Create trip & search
    c. Navigate to the authorization endpoint, and input your API key in the `Basic` field. The code will automatically convert it into an authorization token (no need to worry about this for now). Click Try It! You should get a working response.
    d. Play around with the create trip & search process to familiarize yourself.
3. Replicate the search in code
    a. Now, we will need to replicate this in code. You can do so using the Python Requests library here.
    b. Begin coding in `script.py` (currently empty) so that it interacts with Gordian's API to do the following for each of the trips:
        i. Create a trip and search seat map data for the itinerary.
        ii. Save the seatmap in the /output folder, following the naming convention set above.
4. End deliverables for this first part:
    a. `script.py`
    b. in the `/output` folder, `trip_1.json` —> `trip_4.json`

    c. Libraries you’ve used to complete the task (e.g. requests library) in `requirements.txt`

Task 2: Parse seatmap

Overview

Great work! You now have the outputs to the seatmap, and you understand how the general structure of the seatmaps look. If you were unable to grab a seatmap output, please open up `/output/example_output.json` to take a look.

In this section, you are tasked with creating a parser that

- Parses through a provided XML file of seats & offerings
- Returns the result in our specified output format.

Notes:

- It is important to make sure that the code you produce has a reasonable runtime and memory performance. We are looking for logical and easy-to-follow code flow. We will also be evaluating the various uses of tools in your code. The full rubric is available here.

Steps

1. Understand the problem:
    1. ***Processor***: a python script that processes the xml to json
        i. `/task_2/seat_parser.py`
    2. ***Input***: an unparsed XML seatmap
        i. `/task_2/seatmap.xml`
    3. ***Output***: a parsed json seatmap
        i. `/task_2/seatmap.json`
    4. ***Schema***: output parsed json seatmap should follow the below schema:
        i. `/task_2/seats.yaml`
2. Begin creating `seat_parser.py`. This file is initially empty–you should write your script in here. The script should take in `seatmap.xml`.
3. `seat_parser.py` should write the resulting seatmap to `seatmap.json`
4. `seatmap.json` should follow our standard seatmap structure you observe in previous & example seatmap outputs.
    1. Please omit the following:
        i. in_settlement_currency and in_original_currency fields in seat_prices
        ii. markups and ota_markups

5. End deliverables for this second part:
    1. `seat_parser.py` (script)
    2. `seatmap.json` (output of the formatted xml file)

What's next?

Congratulations on completing this take-home interview! We are excited to read through your deliverables and comments. If you have any comments on the process you'd like to note, please write it in comments.txt.

You can expect a response within 3 days of submitting the materials.

Cheers,

Gordian TPMs