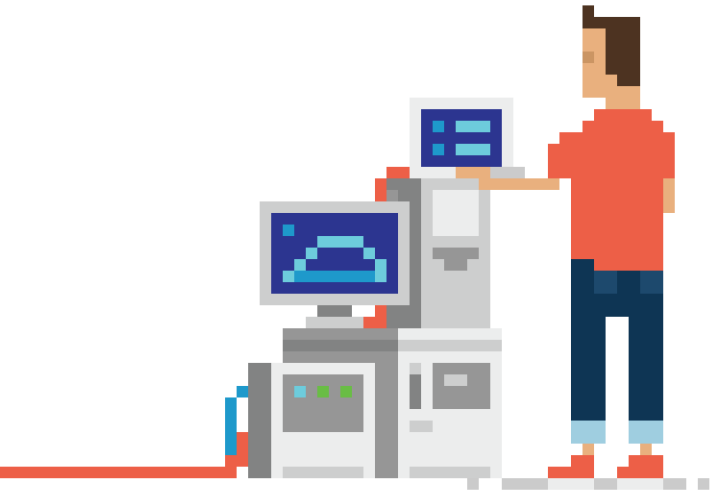

mirumee™
RECRUITMENT
MISSION

Welcome to Mirumee Software's recruitment task for backend developers!

The exercise consists of a set of several smaller steps - the result of each should be included in the final code provided by the candidate. Please feel free to reuse any solution from previous steps to solve the next one. The tasks are designed to give you the possibility to show your coding skills, as well as your general software development knowledge and, are defined in an open manner. Completing the full mission will require at least a few hours, even for a skilled developer!

We don't want to take whole days out of your schedule, so please feel free to establish a time-cap suited to you, eg. four hours, and deliver as much functionality and quality as you can in that time. Our goal is to get to know you as a problem solver and coder, this not a competition. The scope of work should be sound and clear, so if you're unsure about anything, please reach out to us so we can clarify the requirements. If not explicitly mentioned, established industry standards apply.

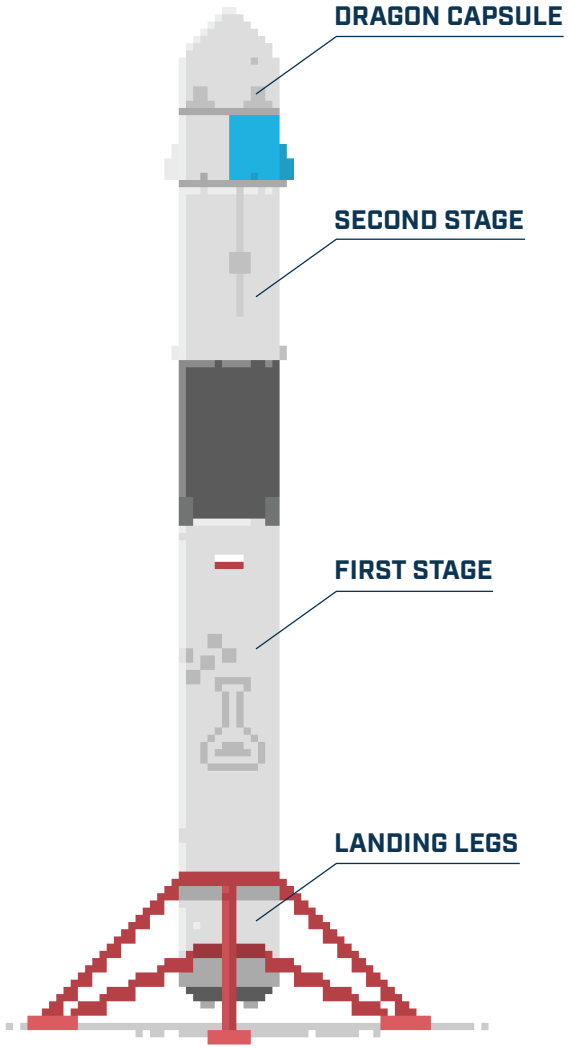


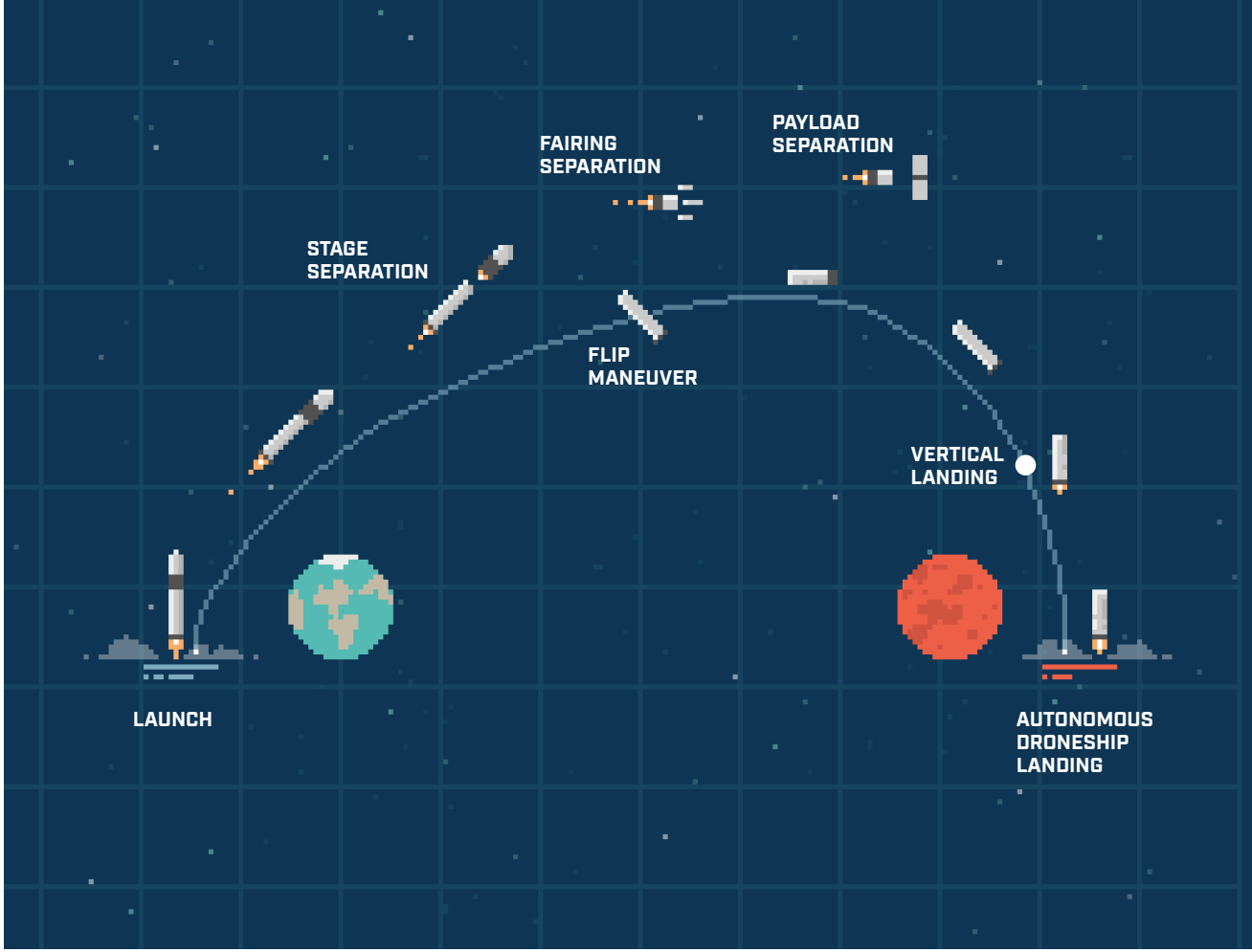
MEET YOUR DATA

At api.spacex.land/graphql you will find a publicly available, community-driven, GraphQL API that serves all kinds of data related to SpaceX operations, such as the usage of different rocket cores, payloads, human flights, etc. It will serve as a convenient data feed for our little programming challenge. Please take time to walk through the API explorer to become familiar with the data.

STAGE I

LAUNCH THE CODE TO SPACE





The Falcon 9 rocket is stacked up from two independent rocket stages and a payload hidden in the fairing for protection. The first stage is capable of reaching the edge of space at ten times the speed of sound with a full load. Moreover, SpaceX has managed to engineer a way to land the stages back on an autonomous ship in the middle of the ocean using controllable fins and retropropulsion and reuse the rocket multiple times.

Your first task is to fetch the most used first stages from the API and calculate the overall payload mass carried into orbit by each core. Provide a parametrized function in a dedicated Python module that fulfills the requirements.

PARAMETERS

- number of most reused rocket cores to fetch
- include or exclude unsuccessful flights
- include or exclude planned future missions

OUTPUT

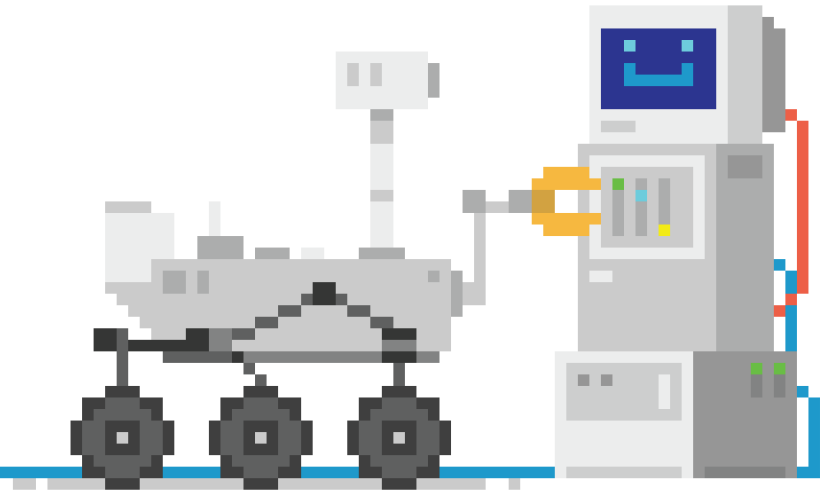
- list [tuple[str, int, int]] which stands for core ID, number of reuse and payload mass carried to space

ADDITIONAL REQUIREMENTS

- code should be tested and unit tests provided
- main.py file provided to execute commands
- code is clear and formatted
- please use as few external libraries as possible, if needed provide a convenient way to set up an environment
- provide a README.md with how to use your program with essential information
- code performance, readability, error handling, documentation, and openness to potential business requirement changes will be taken into account
- the whole project should be provided as a zip file ready for code review and testing
- you may or may not use version control, but if you use it, please keep it simple and easy to follow

STAGE II

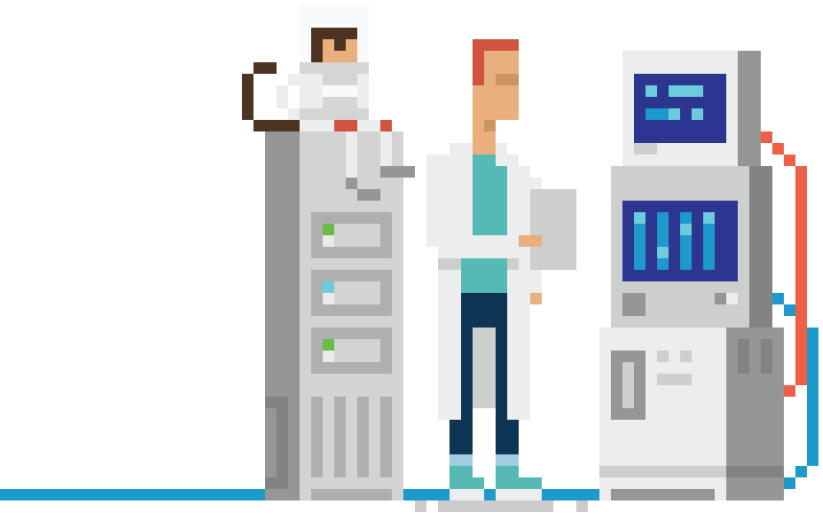
REACH ORBITAL VELOCITY



Great work with the first task, now let's take it to the next level!

In this step, your challenge is to write a simple web app that uses a database to store fetched information and a RESTful or GraphQL API to serve the data.

1. Your database needs to store the rocket core information that has been fetched in the previous step. So, in the core we need: ID, reuse count, payload mass delivered. Create also a relation with the user that consists of the user's name and information about their favorite rocket core.
2. Provide a script/command that creates three sample users for testing purposes.
3. Implement an API endpoint (REST or GraphQL) that provides information about all cores, if the DB is empty, trigger fetching and save to the DB.
4. Add functionality to your API for the user to choose their favorite rocket core and for the user to be able to display this information.



All additional requirements from the first task also apply here.

Additionally, take into consideration:

- adding complementary functions to the API is a bonus
- we know and like the following tools:
  - Django + DRF
  - FastAPI
  - pydantic
  - requests/aiohttp
  - Flask
  - Marshmallow
  - SQLAlchemy
- if you want to use another library that you like, please provide a GitHub/GitLab/BitBucket link and a short explanation about the reasons why you chose it. Perhaps you know it well and use it daily? Just let us know.
- we're not robots, we will look into your code with a human eye so as long as it performs required functionalities feel free to show your personal view on coding, API design, architecture, workflow, etc.

So that's it! We hope you have some fun while playing around with the rockets. Now just attach the zip files to the email and send your results back to us - will get back to you soon.


mirumee™
WELL DONE
THANK YOU FOR PARTICIPATION