TRACKING THE INTERNATIONAL SPACE STATION

Matthew Thomas

COE 332

Spring 2025

Overview and Motivation

The International Space Station is an immensely important piece of technology revolving around Earth that houses astronauts and provides an environment to conduct space research. It is 357 feet wide and has a mass of about 925,000 kg. Because of this, tracking its location and velocity is useful to ensure that it remains free of collisions and is continuing to follow its projected path. It is also interesting to know the geolocation of the ISS because of the potential for it to be over one's city!

This software project consists of an API that strings together a Docker container, Redis database, and Flask application to track the ISS as it travels along its trajectory. The data used in this project comes from the NASA website and is uploaded into a Redis database on startup. The software supports multiple Flask routes that return insights into the geolocation and motion of the ISS. This project demonstrates each of the four software design principles by being portable, accessible, and well organized.

Data

The data used in this project comes from the NASA ISS tracker website, which is linked in the References section. The data is originally in XML format and contains the ephemeris of the ISS every four minutes over a fifteen-day period. This means that the data is updated consistently and specifically contains previous and future location and velocity data for the ISS. To use this data effectively, a list of the "stateVectors" was extracted which contained dictionaries filled with the data of interest. A snippet of the raw data is shown:

```
<stateVector>
<EPOCH>2025-083T10:42:30.000Z</EPOCH>
<X units="km">-943.08541634318203</X>
<Y units="km">-4264.3048850485902</Y>
<Z units="km">-5215.3149701879802</Z>
<X_DOT units="km/s">7.5438109206755</X_DOT>
<Y_DOT units="km/s">-0.13330386059415</Y_DOT>
<Z_DOT units="km/s">-1.25374730378107</Z_DOT>
</stateVector>
```

Software and Services

The following software and services were used to create and deploy this project:

1) Docker:
   Docker is an open-source platform that allows developers to package their programs along with all their dependencies into portable 'containers', ensuring equivalent functionality across all environments. These containers are stored on Dockerhub, where users can pull them to their local machine and run the program in

its entirety. In this project, users pull my public image with its necessary software and Python libraries, from Dockerhub to run my program. In fact, users never have to interact with Flask or Redis directly, as they are both packaged and ran in the container.

2) Flask:
   Flask is a lightweight framework for web development in Python. It provides the functionality to create app routes that can power API's over the internet. In this project, Flask is used to produce web routes that users can curl on their local machine to access the data and the Python functions.

3) Redis:
   Redis is an open-source NoSQL database that stores data in memory. It is primarily used for cache and as the means to build and deploy applications. In this project, Redis will store the ISS data and return queries/specific parts of it as requested by my functions. An advantage of using Redis here is that it allows the data to be continually accessible and persist through runs.

All code for this project was written in Python 3.13.

Usage

Users can pull this project from my public image on Dockerhub, gaining the ability to interact with the Flask server and Redis database that stores the data. When the application is started for the first time, the data is read from the NASA URL and stored in a Redis database in key-value format where the keys are timestamps, and the corresponding full epoch are values. Users can curl the Flask routes on their local machine to run my analysis functions and receive output to their screen. Each route causes a retrieval of data from the Redis database directly. Additionally, the data on Redis persists; if a user were to restart the application, the data would not be retrieved from NASA again.

The following table details the functionality of each app route that can be curled:

APP ROUTE	INPUT	OUTPUT
/epochs	None	All epochs in the dataset, formatted in XML
/epochs_query?limit=<start>&offset=<last>	'start' & 'last' – day of the year bounds for the range of data the user requests	A subset of the dataset
/epochs/<epoch>	'epoch' - The given epoch formatted as it is in the raw data	The state vectors for the epoch
/epochs/<epoch>/speed	'epoch' - The given epoch, formatted as it is in the raw data	The instantaneous speed in km/s of the ISS at that epoch
/now	None	The current epoch, latitude, longitude, altitude, and geolocation of the ISS
/epochs/<epoch>location	'epoch' - The given epoch, formatted as it is in the raw data	The latitude, longitude, altitude, and geolocation of the ISS at that epoch

All code uses defensive programming strategies to increase the simplicity of its use and accommodate less-savvy users. An example of such a method is shown.

```
def find_epoch(epoch : str) -> list:
    '''
        This function returns a specific epoch from the dataset
        Args:
            epoch: an epoch (timestamp)
        Returns:
            i (list of dicts): the full epoch given
    '''
    pattern = r'^\d{4}-\d{3}T\d{2}:\d{2}:\d{2}\.\d{3}Z$' # epoch format (used ChatGPT to
generate)
    if not re.match(pattern,epoch): # error if epoch not in correct format
        return 'Invalid epoch entered'
    try:
        return json.loads(rd.get(epoch).decode('utf-8'))
    except AttributeError:
        logging.warning('Epoch is not contained in the current ephemeris')
        return 'Epoch is not contained in the current ephemeris'
```

If a user inputs an invalid epoch or one that is not contained in the current dataset, the error is handled gracefully, and the user is informed.

Ethical and Professional Responsibilities in Engineering

This project promotes the use of ethically sourced and honest data. Since the data used is reported by NASA and about the ISS, we can infer that it is highly accurate and was produced by responsible engineers with their consent. In an age of both dwindling data privacy and AI, it is imperative to uplift reputable data sources, like NASA, and only work on data that can be trusted. In addition, it is important that developers take responsibility in how they use services that operate over the internet, such as Flask. All GET requests were done in good faith and communicated with other web servers without attempting to crash them. Lastly, this project demonstrates intellectual discovery and scientific growth. Through gaining a deeper understanding of the ISS, an international space tool, this work has implications that can make society a better place, which is a core tenet of good engineering.

Discussion

Although the raw data required some wrangling to be able to iterate through effectively, once in a clean form, it produced valuable insights into the ISS. The analysis of the ISS data showed that the station typically travels around 7.6 km/s at an altitude of 418-420 km above the Earth's surface. Due to its trajectory, the ISS spends most of its time over oceans and follows a strict path across each continent. The current limitations of this project include the potential for maintaining untimely data. After the program fetches the current ephemeris data from NASA on its first start-up, it will store that data in a Redis database and never update it so long as the database is occupied. Ideally, the program always uses the most current data, especially for time sensitive routes such as '/now', to avoid analyzing data that does not showcase the most relevant fifteen-day period. However, this problem can be mitigated by users flushing the Redis database with the Redis 'flushall' command before every run, especially if there has been substantial time since the last run. This will ensure that the ISS data is fetched and stored on startup.

Overall, creating this project was a positive experience and exists as a wonderful demonstration of combining multiple services and technologies to produce one, finished product.

References

https://spotthestation.nasa.gov/trajectory_data.cfm

The raw ISS data can be found at this link.

https://geopy.readthedocs.io/en/stable/#module-geopy.geocoders

The GeoPy documentation guide provided insight into how to use its functions to find the geolocation of the ISS.

ChatGPT

ChatGPT was used for general code and software debugging. Lines of code generated by AI are annotated within the source code.