International Space Station (ISS) Trajectory Tracker

Arushi R. Sadam, UT EID: ars7724

08 March 2024

COE 332: Software Engineering & Design, Midterm Project

Professors: Dr. William Allen, Professor Sushobhon Dey, Dr. Joe Stubbs

Table of Contents:

1. Purpose:

The purpose of this project is to predict and track the International Space Station's (ISS's) position within a variable time range. Based on the ISS's position, velocity, altitude, and geoposition (above which city the ISS is traveling), this project could be extended to provide insights into orbital dynamics, atmospheric drag, and the gravitational effects of Earth and other celestial bodies. Additionally, this project could help avoid accidental collisions involving the ISS and other spacecraft, ensuring the safety and security of the ISS, spaceships/rockets, satellites, and respective crews. Finally, for astronomers or individuals interested in viewing the ISS, this project allows easy access to the ISS's geoposition at any given time, enabling ISS sightings.

2. Data Source:

The ISS Trajectory Operations and Planning Officer (TOPO) flight controllers in the Mission Control Center at NASA's Johnson Space Center generate the ISS trajectory data [1]. Data includes the ISS's position (x, y, and z coordinates) and the ISS's velocity (also given in x, y, and z coordinates) across 15 days at intervals of approximately 4 minutes; at each data point, the time stamp (in Coordinated Universal Time or UTC zone) is also provided. General information about ISS and the data collection (center name/earth, reference

frame, time frame, start/stop time, etc.) are provided in the header and metadata. Comments about the data units, ISS mass, drag coefficients, etc. are also provided.

[1] provides a text (.txt) and an XML (Extensible Markup Language, .xml) data file. This app uses the XML data format.

3. Program:

This program uses Docker as a platform to deliver the software (written in Python programming language) in a package called a container. The program has been containerized and a template (called an image) has been uploaded to the web. Users can "pull" the image from the web to their local desktops (if they have Docker installed) and then run the program. This method of running the program ensures all dependencies (i.e. Python libraries or specific required Python versions) are installed. Specific Python3 package requirements can be found in the requirements.txt file in [2]. For more information about installing Docker locally, view [3].

To build the web application, this program (or application programming interface, API) uses Flask [4], a Python web framework. Flask allows for easily developed and accessible applications. To run the application, users simply need to use a docker run command to run the API. Then, with Flask, they can use quick curl [5] (client for URL) commands to interact with the running API and get different outputs displayed to the terminal.

The program (GitHub Link for access to files: [2]) is made up of seven files:

1. "iss_tracker.py": code that requests data about the ISS's position/velocity from the NASA public website [1] based on user's flask commands.
2. "test_iss_tracker.py": contains unit tests for iss_tracker.py.

3. "requirements.txt": the required versions of python libraries (used in Dockerfile).
4. "Dockerfile": contains instructions for docker to work (building/running program).
5. "docker-compose.yml": containerized docker commands (for automation purposes).
6. "diagram.png": diagram of program (shows user-app interaction).
7. "README.md": describes functionality of app (includes details on how to run specific commands on the app).

The 7 files and their interactions are shown below in Figure 1:

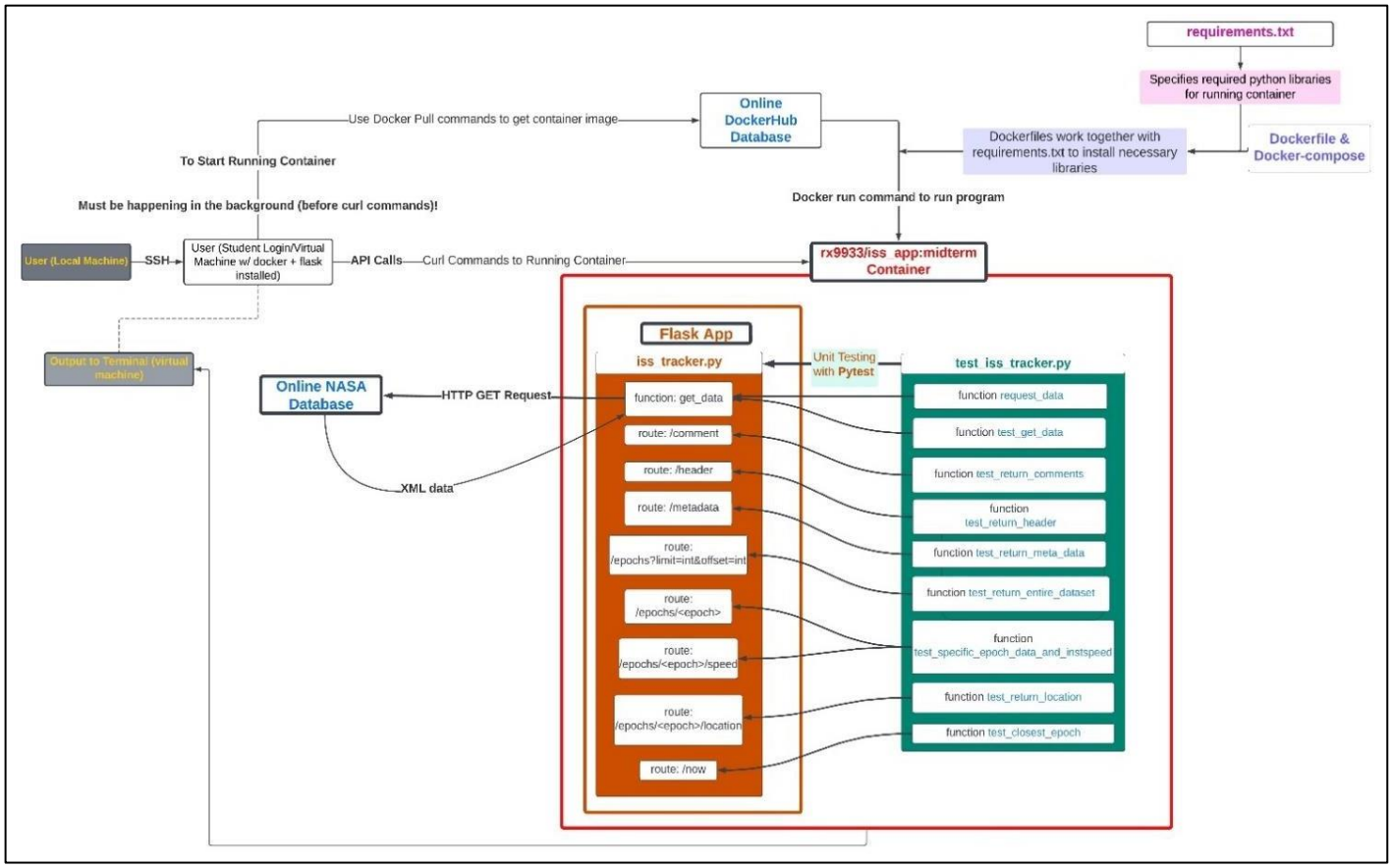

Figure 1: Software Diagram

Note: while reading through the write-up, please refer back to the software diagram for more details.

Ultimately, this program provides the following ten functionalities (Figure 2):

Route	What it should do
`/comment`	Return 'comment' list object from ISS data
`/header`	Return 'header' dict object from ISS data
`/metadata`	Return 'metadata' dict object from ISS data
`/epochs`	Return entire data set
`/epochs?limit=int&offset=int`	Return modified list of Epochs given query parameters
`/epochs/<epoch>`	Return state vectors for a specific Epoch from the data set
`/epochs/<epoch>/speed`	Return instantaneous speed for a specific Epoch in the data set (math required!)
`/epochs/<epoch>/location`	Return latitude, longitude, altitude, and geoposition for a specific Epoch in the data set (math required!)
`/now`	Return instantaneous speed, latitude, longitude, altitude, and geoposition for the Epoch that is nearest in time

Figure 2: Routes for API [6]

Each route represents a unique addition to a standard curl command, and the corresponding "what should it do" explains what will be output to the terminal.

After users run the API (with a docker run command), they can use a basic curl command (curl localhost:5000/) followed by a specific route to return the corresponding output to the terminal. For example, to return a comment list object from ISS data (Figure 2, first route), users can use the following command: curl localhost:5000/comment. If users wish to see the ISS data (position and velocity) between two time periods, they can use: curl "localhost:5000/epochs?limit=2&offset=200". This would return a total of two data points (since limit = 2) starting from the 200th data point in the data set (since offset = 200). Note the quotation marks in this command; when providing more than one query parameter—input

values (i.e., limit and offset)—users must remember to use quotes for accurate data processing. Details about running the API and using specific curl commands can be found in the README.md file in [2].

4. Ethical and Professional Responsibilities:

Tracking the ISS's position raises ethical concerns regarding privacy and data security; users must responsibly use such tracking technology. Immediate access to the ISS's location (which city it is over) in a user-friendly manner could potentially threaten the privacy and security of ISS astronauts. For professional apps/software that rely on the ISS's position to navigate spaceship docking with the ISS, inaccurate locational data could jeopardize the mission, imperiling astronauts or crew and wasting millions of dollars. Citing sources/data allows for verification of data, thus potentially preventing any mistakes/errors. Additionally, ensuring accurate descriptions (i.e., that time epochs are provided in the Coordinated Universal Time/ UTC time zone) is crucial for successful space missions; this information can also be verified if accurate data citations are provided.

5. Citations:

[1] “Spot the Station,” Nasa.gov, 2018. https://spotthestation.nasa.gov/(accessed Mar. 01, 2024).

[2] A. Sadam, GitHub. https://github.com/rx9933/ISS-Tracker (accessed Mar. 01, 2024).

[3] Docker, “Enterprise Application Container Platform | Docker,” Docker, 2018. https://www.docker.com/

[4] “Welcome to Flask — Flask Documentation (3.0.x),” flask.palletsprojects.com. https://flask.palletsprojects.com/en/3.0.x/

[5] “curl - How to Use,” curl.se. https://curl.se/docs/manpage.html

[6] J. Allen, “Midterm Project — COE 332: Software Engineering & Design documentation,” coe-332-sp24.readthedocs.io. https://coe-332-sp24.readthedocs.io/en/latest/homework/midterm.html (accessed Mar. 02, 2024).