In Your Orbit

Requirements Document

Version 1.2

Cloud-Based Planetary Ephemerides

United States Geological Survey
Kelvin Rodriguez
Amy Stamile
Christine Kim
Adam Paquette

Mentor: Scott Larocca

Austin Carlile
Noah Schwartz
Minuka Trikawalagoda
Nicholas Gonzalez

The purpose of this requirements document is to detail the functional, environmental, and performance requirements of the solution, as well as the potential risks.

11/12/2024

Client Signature Date

Team Lead Signature Date

Table of Contents

1. Introduction

The field of planetary science, particularly NASA missions, as a billion-dollar industry, is immensely important and growing rapidly as new technologies emerge. The processing of planetary images is quite critical for research and exploration and the sensor models that process this information play a key role. These sensor models convert coordinates into spatial reference systems, i.e., longitude and latitude, allowing scientists to reconstruct planetary topography and add geo-locations to imagery with high accuracy. The data from the models support various NASA missions including the upcoming Europa Clipper mission, the aim of which is to investigate potential life on one of Jupiter's moons.

Currently, the biggest challenge lies in the reliability of retrieving Image Support Data (ISD), which is essential for creating the aforementioned sensor models. The ISD has various parameters in its programming that allow for unique identification, however, generating ISD involves navigating NASA's enormous SPICE data system which presents a significant challenge for new users. As more NASA missions commence, the SPICE data system grows almost unmanageable, making it even more important to simplify the process.

In an effort to streamline the process of caching and retrieving ISD from the SPICE database, our capstone team will develop a cloud-based web service that builds on top of existing USGS software. Utilizing various AWS services, the service will allow users to retrieve ISD without needing the entirety–up to multiple terabytes–of the SPICE database. Our goal is to create a more user-friendly service that is more accessible to new scientists who are trying to emerge into the planetary science field by shifting their efforts from data management to scientific inquiry.

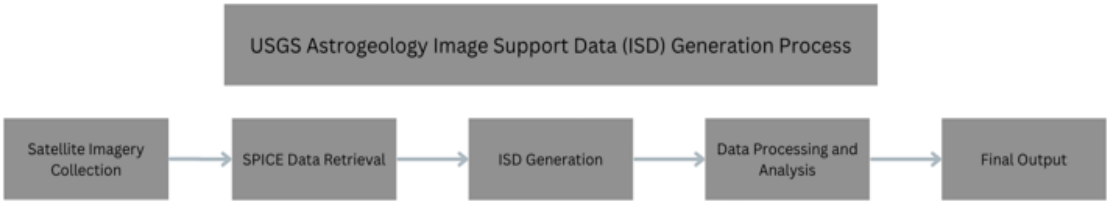
The sponsors of this project—Kelvin Rodriguez, Amy Stamile, Christine Kim, and Adam Paquette—are computer scientists at the USGS Astrogeology Center. All four scientists are well-versed in the ins and outs of planetary science and data positioning, making them qualified to guide us in the development of this project. USGS Astrogeology plays an important role in advancing our understanding of planetary bodies as well as providing vital support for various

NASA missions. By streamlining the process of retrieving ISD, we will greatly contribute to their efforts in conducting research efficiently and effectively.

Success in this project will enable both USGS and NASA scientists to process planetary imagery on a large scale, facilitating more efficient on-premise workflow but cloud-based as well. Ultimately, this project aims to lower the barriers for future research students in planetary science, promoting greater accessibility and innovation in the field.

2. Problem Statement

USGS Astrogeology is a crucial part of planetary science, offering satellite image data that is essential to planetary mapping for mission analysis. Currently, ISD generation is costing USGS over $10,000 a month and is bulky, slow, and not at its peak of performance. Our proposed solution will consist of a new way to return ISDs by establishing a new serial code system, which will return ISDs with less searching. Additionally, it will compress data so that users are no longer downloading data sets that are over a terabyte. The existing workflow's dependency on the SPICE system results in slow, complex ISD generation. The growing volume of SPICE data and lack of streamlined access mechanisms make it increasingly difficult for researchers to retrieve and utilize the data effectively, stalling critical mission support functions.



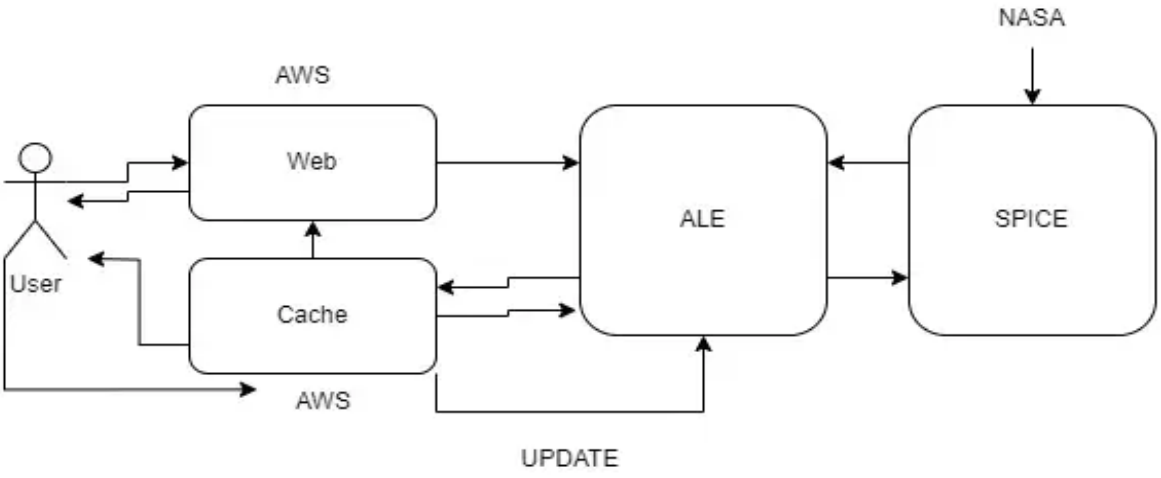
Above is our client's current process to generate an ISD, with the main problems occurring within the third and fourth tier. When ISDs are generated they have certain tags to identify them (e.g. mission, start_time, instrument), however, the current system does not develop efficient identifiers, leading to long search times, and large amounts of storage required, which ends up

costing USGS large sums of money. Additionally, the datasets involved can be as large as a terabyte, making ISD generation on smaller machines quite difficult.

Current Deficiencies

- Data sets can be too large for small machines to utilize
- Inefficient identifying keys causes slow search times which become costly for the company
- ISD generation is time-consuming for scientists and researchers which can lead to delays in missions
- Currently, there is no good interface to facilitate ISD requests, leaving users to invest time into data handling instead of research

3. Solution Vision



The project seeks to create a cloud-based service that provides efficient and streamlined access to Image Support Data (ISD) for planetary scientists, addressing current data retrieval challenges from NASA's SPICE system. This solution will improve ISD generation speed, accessibility, and resource efficiency, enhancing research workflows and supporting NASA's mission objectives.

Key Components

1. **ISD Generation and Caching**
    - The system will generate ISD files based on user-specified parameters, utilizing ALE, a NASA tool for planetary data.
    - ISD files will be compressed in JSON format to minimize data size, making data transfer and storage more efficient.
    - Frequently requested ISDs will be stored in Amazon DynamoDB, enabling quick retrieval and significantly reducing response times for repeat requests.
2. **Dynamic Scaling and Performance Optimization**
    - The service will be deployed on AWS ECS, with autoscaling via EC2 to handle up to 200,000 simultaneous requests.
    - This scalable infrastructure will ensure consistent performance, even under high user demand, supporting large-scale scientific and mission-planning activities without compromising speed.
3. **Automated Cache Management**
    - The caching server will monitor updates to the SPICE database and automatically invalidate outdated ISD entries.
    - This ensures that users always access the most current data without manual updates, maintaining data accuracy and reducing user burden.
4. **User-Friendly, Queryable Interface**
    - The web service will offer a straightforward, accessible interface that allows scientists to query for ISD data quickly and easily.
    - This design lowers entry barriers for new scientists in planetary science, enabling them to focus on research rather than data handling, and enhances productivity for experienced researchers.

Anticipated Benefits

- **Reduced Data Access Costs**: By optimizing ISD retrieval and minimizing large data downloads, the solution will reduce storage and access costs significantly for USGS.

- **Enhanced Efficiency and Speed**: Cached data and compression strategies will shorten ISD generation time, allowing researchers to access data faster and with less overhead.
- **Scalable, Reliable Service**: AWS integration enables dynamic scaling to meet varying demands, ensuring the service remains robust and efficient for high-volume scientific use.
- **Improved Accessibility for New Scientists:** The user-centered design of the service will lower entry barriers, making planetary science more accessible to new researchers and fostering innovation in the field.

4. Project Requirements

The solution needs a universally accessible, online-hosted web service that can be easily queried for the latest ISD Images. To accomplish this task, the solution must have the following Functional, Performance and Environmental requirements:

4.1 Functional Requirements

In order to deliver a successful solution to the client, there are a few functions that our application needs to implement. The following is a detailed list of every functional requirement that our solution must have:

4.1.1 Web service

Our solution will require us to create a web service that can generate ISDs or request them from a connected caching server and return them to the user.

1) **Hosted on Amazon AWS**

   This web service will need to be hosted on Amazon AWS services in order to be accessible, portable, and scalable.

a) **Use ECS container**

To host the Web Service and utilize Amazon AWS services, the application will sit inside an Amazon ECS Container that allows the pairing of AWS services to handle various responsibilities of the system.

b) **Host web service on EC2 Instance**

The Web Service itself will be hosted on an Amazon EC2 Instance that is contained in the ECS container.

c) **Use EC2 autoscaling for scaling**

In order to scale the Web Service based on demand, Amazon EC2 Autoscaling will be used to dynamically scale the EC2 instances and efficiently respond to demand.

2) **Allow user to query for ISD**

The web service will provide an easy to use queryable interface for searching for ISDs from the NASA SPICE database. The web service will request the ISD from the caching server and return the ISD to the user. If the ISD is not stored in the caching server, the web service will generate the ISD using ALE and both send it to the caching server to be stored, as well as return it to the user.

a) **Search SPICE to find relevant data**

If the ISD is not in the caching server, the Web service will have to search the SPICE database for ISD source data.

b) **Generate ISD**

The Web Service will use ALE to generate ISDs, compress the ISD to minimum size using binary conversion and bit compression. It will also generate a custom ID to identify specific ISDs in the caching server.

c) **Communicate with the caching server**

Connection to Caching server, Send generated ISD's to the caching server. Retrieve the latest ISD from the caching server, Request the latest ISD for the user using identifying information.

d) **Return ISD to the user**

The Web Service needs to return the ISD to the user, whether it was generated or retrieved from the caching server.

4.1.2 Caching Server

The caching server will store all previously requested ISDs and update them if there are changes to the source SPICE kernels, as well as returning requested ISDs to the web service. The caching server will also utilize a custom ID numbering system that will maintain the version of each ISD and will only return the latest ISD version to the web service when requested.

1) **Hosted on Amazon AWS**

This web service will need to be hosted on Amazon AWS services in order to be accessible, portable, and scalable.

a) **Use ECS**

The caching server will make use of Amazon DynamoDB to create an Object relational database that will store ISDs and keep track of their versioning history and what SPICE source kernels they came from.

2) **Communicate with web service**

The caching server will need to communicate with the web service in order to receive and return generated ISDs.

a) **Connection to web service**

The caching server will create a TCP connection to the web service that will utilize RESTful service using FastAPI to implement.

3) **Store ISD**

When a new ISD is given to the caching server from the web service, the caching server will need to store the ISD with a custom ID

a) **Use a custom ID number for ISD**

The caching server will need to have a mechanism to generate a unique ID for every ISD stored. This will exist in a local script that is stored with the database on the caching server.

b) **Store updated versions of ISD**

When an updated ISD has been generated, the caching server will use a modified ID number identifying which version of a specific ISD it is. The caching server will also store the generated ISD alongside the original ISD in order to keep a history of all changes.

4) **Return the latest ISD to the web service**

When the web service requests an ISD from the caching server, it will return the latest version of the ISD to the web service.

a) **Find the latest version of the requested ISD**

The caching server will use the ID number to find the latest version of ISD by identifying the ISD ID that indicates the highest version.

5) **Check ISIS for kernel updates**

Periodically, the caching server will run a script that will check all ISD sources for changes. This process will use the source location information stored alongside the ISDs in order to efficiently search the SPICE database for the relevant data.

a) **Search kernels for changes**

Keep track of which kernels ISD images came from, Keep a table or column for kernel sources for each ISD. Periodically Iterate through table or column, Run

scheduled task, Check all kernel sources for changes, Update ISD if the kernel has changed.

6) **Update ISDs**

   If the caching server detects any changes in the ISD source kernels, it will request that the web service generates a new ISD and will store it using a modified ID of the outdated ISD.

   a) **Use web service**

      The caching server will request an ISD to be generated by the web service and returned to the caching server to store

   b) **Store ISD using incremented ID**

      The caching server will store the ISD using an incremented version of the outdated ISDs ID number

4.2 Performance Requirements

In order to deliver a successful solution, the application will have to perform to certain standards. The following is a detailed list of every performance requirement that our application must achieve:

1) **Scale to two hundred thousand requests**

   Our solution will need to be able to handle up to two hundred thousand simultaneous requests. In order to achieve this performance requirement, The web service will need to dynamically scale to handle all potential traffic.

2) **Operate at zero cost**

   The solution that we hand to the client must be able to implement all functional requirements and operate entirely for free.

3) **Improve the speed of ISD generation**

   In order to be a successful solution, it will need to improve the speed of the process of generating ISDs for the user.

4.3 Environmental Requirements

The solution must be developed with a couple restrictions. The following is a detailed list of all Environmental requirements that our application follow:

1) **Written in Python**

   The web service must be coded entirely in the Python programming language. It will need to utilize FastAPI and follow RESTful design philosophy.

2) **Using Amazon AWS services only**

   The solution must only use Amazon AWS services to handle all of the backend of the application. The web service must sit on an AWS server and the caching server must be an AWS database, all external functional and performance requirements will need to utilize AWS services to implement them.

5. Potential Risks

When analyzing our project in terms of risk, there are two main categories, performance and accuracy.

5.1 Performance Risks

The main performance risk that our Web Service solution faces is the risk of inefficient application scaling. To tackle this problem, we will develop our application in a way that allows easy separation of components, primarily separating the Web Service from the Caching Server. This architecture will allow us to utilize Amazon scaling services for individual aspects of the application and provide us with an efficient method of automatically scaling our solution. The impact of this risk is moderate, as inefficient application scaling will not have catastrophic

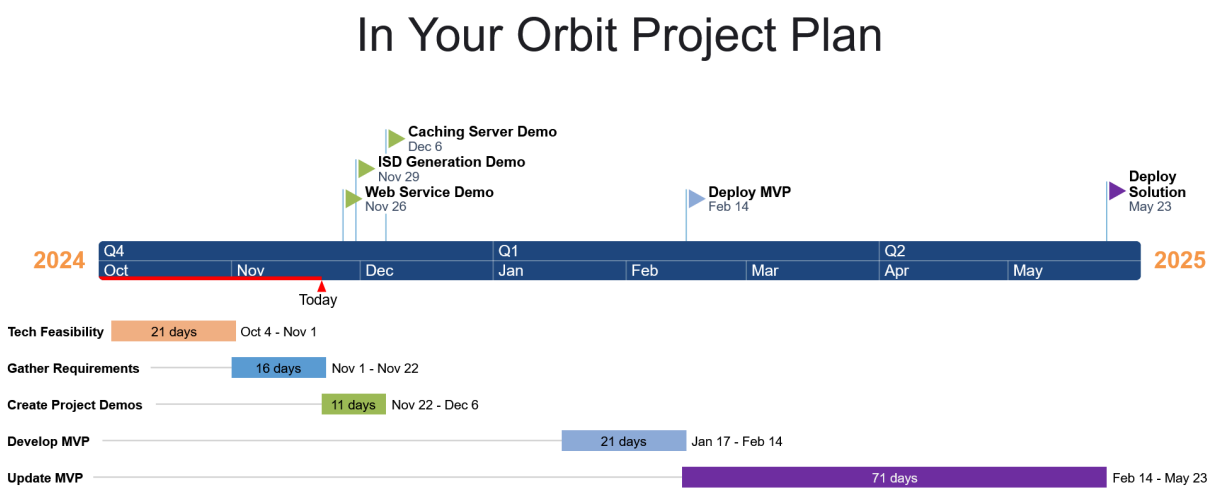
consequences for the users but will make our solution less effective for its purpose. While a more efficient competitor could make our solution obsolete, it will provide a strong foundation that can be upgraded if needed.

5.2 Accuracy Risks

The main accuracy risk of our Web Service solution is accidentally generating or returning the wrong ISD data to the user. To tackle this problem, we will be using verification mechanisms to verify that ISDs are generated correctly. The web service will go through the process of generating a requested ISD twice and compare both ISDs against each other, keeping the first ISD if both are equivalent, or discarding both ISDs and beginning the process again if they are not. This will ensure that every ISD is accurate and generated correctly. The impact of this risk is high, as inaccurate ISD generation and retrieval will return inaccurate data to the users and will make our solution completely ineffective. An alternative product that generates correct ISDs and returns them to the user would make our solution obsolete. However, even if our solution's ISD related functionalities are inaccurate, we will be providing the important infrastructure of a working solution that can be easily modified to correct accuracy issues.

6. Project Plan

This document marks the completion of the requirements gathering part of our solution development. Moving forward, we will begin to tackle the functional requirements and risks that we have outlined here by developing demos of the main features of the web service.



In the project plan diagram above, you can see that we are about to begin the process of developing functional demos of our solution. Our main technology demo that we will be developing and deploying is a demo of our ISD generation capability. This will showcase our solutions ability to correctly and accurately generate an ISD using ALE. Technology demos of the Web Service and Caching Server are stretch goals that we will achieve after our ISD generation demo, if time constraints allow. After we have completed these demos detailing our functional solution, we will begin working on developing a Minimum Viable Product. After completing and deploying the MVP, we will be iterating and updating it until we achieve complete required functionality and deploy our final working solution.

7. Conclusion

The current system that Planetary Scientists use to generate ISDs is inefficient and involves downloading and navigating large sets of data from NASA's continuously growing SPICE data system. In order to make this process simpler and more efficient, we are developing a Web Service and Caching Server for USGS and NASA that will allow the easy generation and querying of ISD data. Our solution will need to generate, store, retrieve, and update ISD data in an easy-to-use, queryable web service. Our potential risks are both performance and accuracy related, with risks in application scaling and ISD verification. The purpose of this requirements document is to detail the functional, performance, and environmental requirements of our

solution. This document also details the risks involved and how we plan to mitigate and solve them. We have made considerable progress in the planning stages of this application and are on track to create our technology demos showcasing the viability and functionality of our proposed solution. After that, we will have a strong foundation to begin developing our fully functional and working solution.

8. Glossary

ALE - Abstraction Layer for Ephemerides

AWS - Amazon Web Services

EC2 - Elastic Compute Cloud

ISD - Image Support Data

ISIS - Integrated Software for Imagers and Spectrometers

MVP - Minimum Viable Product

SPICE - Spacecraft Planet Instrument C-matrix Events