Software Design Document

2/18/2022

GeoSTAC

Sponsor:
United States Geological Survey (USGS)
Astrogeology Science Center

Mentor: Melissa Rose

Team Members:
Jacob Cain
Zachary Kaufman
Gavin Nelson
Amy Stamile

Version 2.0

Table of Contents

1 Introduction

New technological advances have opened opportunities for space research and exploration[1]. The majority of this exploration is through launching satellites that orbit planetary bodies, including planets, moons, and asteroids. These satellites collect large amounts of data and take images while orbiting the planetary bodies. The information is then sent back to Earth for the planetary science community to research and better understand our solar system.

As space exploration increases among both federal agencies and private citizens, there is a need for community access to accurate planetary maps and data. Mars is a target for exploration due to its proximity to Earth inside of our solar system. In order to plan for future exploration, it is vital that scientists use the data and images that our satellites and non-human space missions have gathered.

Using these resources, scientists can perform analyses of data from Mars to plan for future missions and scientific discovery. There are many tools that allow scientists to analyze and create maps from the information gathered; however, these tools are not well-developed. The planetary science community processes these images by using software that requires extensive knowledge for complex tools. In addition to using traditional software, the planetary science community must store terabytes of data on their own devices in order to interact with and research the images.

Rather than require researchers to independently process their own cartographic data products, the United States Geological Survey (USGS) Astrogeology Science Center (ASC) in Flagstaff Arizona provides the international planetary science community with analysis ready data. These products support research in planetary cartography, geoscience, and remote sensing. The ASC also develops software for scientific and cartographic analysis of planetary data which can be easily accessed by members of the international scientific community.

1.1 Problem Statement

The USGS Astrogeology Science Center serves data using a community developed standard to the planetary science community to access Analysis Ready Data (ARD). USGS has already distributed this analysis ready data through the Spatial Temporal Asset Catalog (STAC). STAC is a standard for storing, discovering, and

[1] https://www.nasa.gov/specials/60counting/future.html

analyzing spatial-temporal data to describe various geospatial information. This provides better indexing and discovering of the analysis ready data[2].

The STAC specification describes a JSON schema for machine data discovery. The STAC specification also provides an API specification for developers to write human-usable discovery tools. Using these machine-accessible specifications, USGS would like a human-usable data discovery tool that uses web mapping to help users search for and locate data.

USGS needs an interactive visualization tool to link the analysis ready data from the STAC API. This will allow the scientific community to make discoveries of planetary data and further facilitate interactions with the STAC API.

In 2019, the USGS assigned an NAU capstone team to develop an interactive web map that supports planetary data. The map is called CartoCosmos, reflecting the capstone team's name. The CartoCosmos team developed a plugin extension for Leaflet, an open source Javascript library for interactive maps. This plugin extension was developed to support mapping of planetary data sets.[3] The CartoCosmos web map is great for visualizing large planetary image mosaics, but has no support for visualizing the individual STAC assets or STAC catalogs the clients wish to make available.

Thus, the USGS team has tasked Team GeoSTAC with upgrading and adding new features to the interactive web map. This includes the ability to visualize individual STAC asset locations on a map of a target body and load the associated images into the CartoCosmos web map.

1.2 Solution Vision

To solve the problem of interacting with the USGS STAC catalog within the CartoCosmos web map, the team will use the following solutions:

- Query the STAC API endpoint for a given planetary body and render footprints onto the CartoCosmos web map.

- Implement a frontend search functionality that queries information from the STAC API and displays the information of the selected footprints.

- Render Cloud Optimized GeoTIFF (COG) images within CartoCosmos for viewing without the need to download large image data.

[2] https://stacspec.org/
[3] https://ceias.nau.edu/capstone/projects/CS/2020/CartoCosmos-S20/#/

The overall vision for this project is to have CartoCosmos display a certain number of footprints for specified planetary bodies. These footprints will be interactive: The user will be able to individually click a footprint and display the information associated with it. The user will have the option to view the COG image associated with the footprint. In addition, the user will be able to select multiple footprints. This will display a table of available footprints of the selected area and allow the user to select a footprint based on the table. Finally, the user will be able to use a searching tool that will allow them to search within the existing rendered footprints to display only the footprints associated with the information searched.

2 Implementation Overview

The Client wants to make their library of STAC data on extraterrestrial bodies available through an easy to use graphical interface. With the application, users must be able to navigate through planetary maps to find STAC images of interest. The first part of the project, navigation through planetary maps, is a continuation of a previous capstone project. Therefore, the Team GeoSTAC will continue to build on the technologies used by the previous team, CartoCosmos by integrating the CartoCosmos planetary mapping application with the client's STAC API.

The map portion of the interface will be developed in Leaflet (a Javascript library for interactive maps), using the CartoCosmos teams' plugin to render extraterrestrial bodies. The Leaflet community has a variety of plugins which allow easy integration into Leaflet. For user interface elements with pre-existing plugins, the elements can simply be added into the app's Leaflet instance. The team's custom additions to the user interface will be written in React JS with components from Material UI, continuing the theme designed by team CartoCosmos. The backend ("Astro") will be expanded by Team GeoSTAC using Leaflet and Javascript, and calls to the client's STAC API will be integrated.

Node.js will be used to manage the environment for the web app. The team will install any needed dependencies with Node, which ensures that anyone who wishes to host a copy of the app will also be able to easily install its dependencies. The web app will be compiled with Babel JS. The Babel compiler allows the app to be written in modern Javascript, but outputs an app in Javascript with better backwards and cross-browser compatibility. The output will be a web app that anyone can open and use instantly in their browser, without any need for setup on the user's end.

3 Architectural Overview

Designing this application requires a comprehensive overview of how the existing CartoCosmos system is built and what the system supports. With this in depth knowledge of the existing system, the team is able to design an architecture that will surround the existing project and build upon it to provide a concrete architectural design of the application.

To provide a high level overview of the software's architecture, the following two high level diagrams will be used to depict both the original CartoCosmos architectural design in Figure 1, and team GeoSTAC's architectural design in Figure 2.

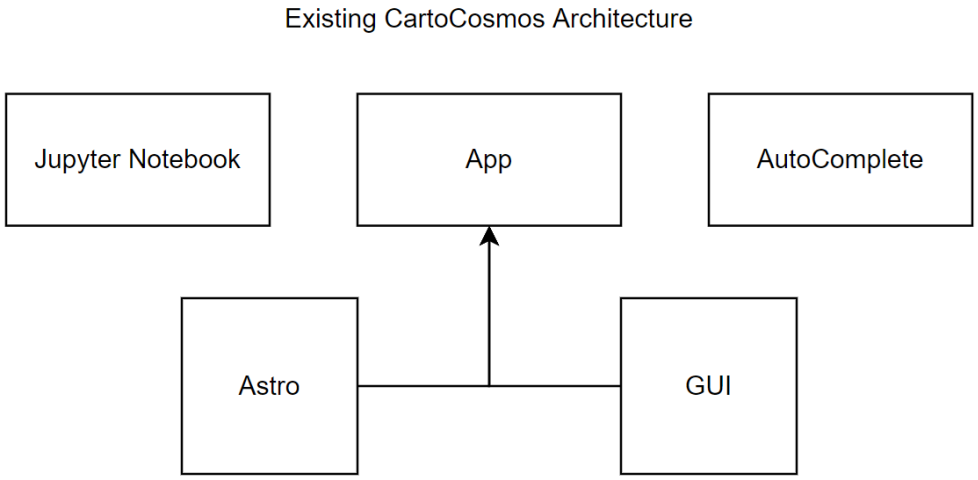

Figure 1: CartoCosmos Architecture

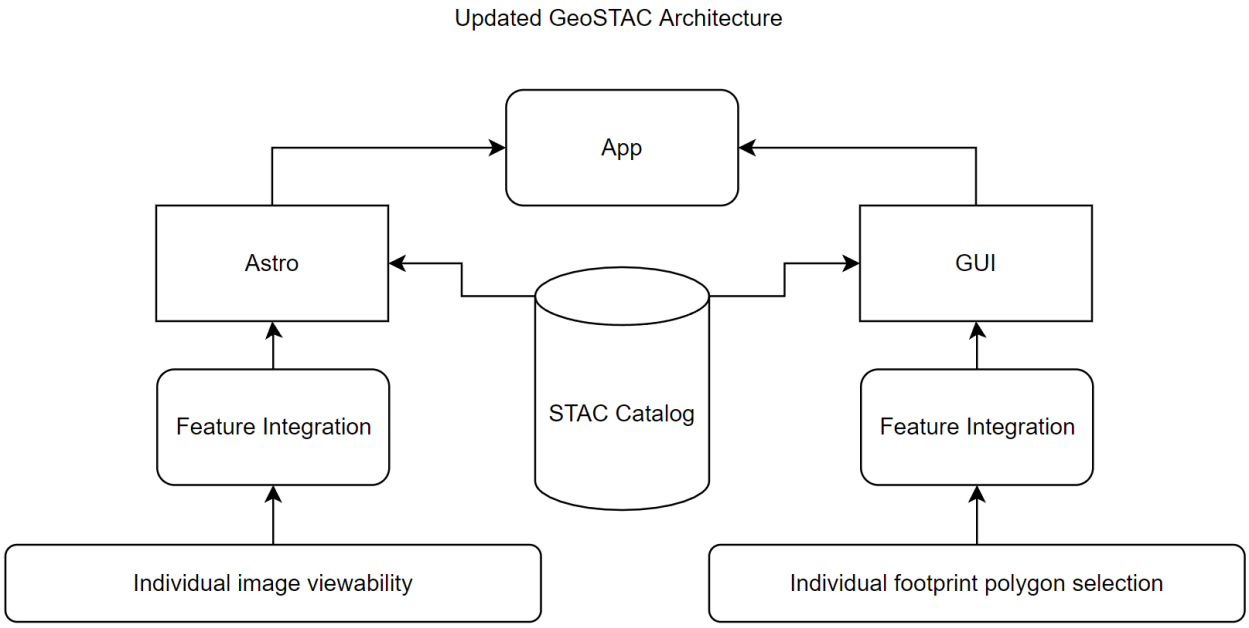

Figure 2: GeoSTAC Architecture

3.1 Astro

As seen from the diagrams above, GeoSTAC is taking the existing architecture from CartoCosmos and building upon the Astro module. The Astro module will be primarily responsible for handling all of the back end interaction. This encompasses API calls, Leaflet map control, catalog selections, and footprint rendering. This module will contain collections of code structured purely around Leaflet as well as vanilla Javascript along with following an OPP approach.

The functionality of using a STAC Catalog has been added into the Astro module as well as added support for interacting with individual footprints. With the additions to CartoCosmos' Astro module, this will provide the architectural framework needed to support the use of STAC Catalog's in interactive Spatial Body Leaflet maps.

3.2 GUI

The existing GUI module has been modified in order to support the functionality of selecting individual footprints that have been rendered on the map. The changes to this module help support map selection of footprints and allow filtering and sorting of footprints. The filters will be based on querying the API on various features that pertain

to the footprints; such as bounding box (bbox) values, time, pages, and other queries that the API supports.

4 Module and Interface Descriptions

The following sections go into details regarding what functions and classes have been added to the existing Astro and GUI modules. Figure 3 displays what existing functions/classes that exist in CartCosmos alongside newly added functions/classes added by GeoSTAC. These new additions are highlighted in blue to provide a visual as to what is being added to existing classes and what new classes are being created in the Astro module.

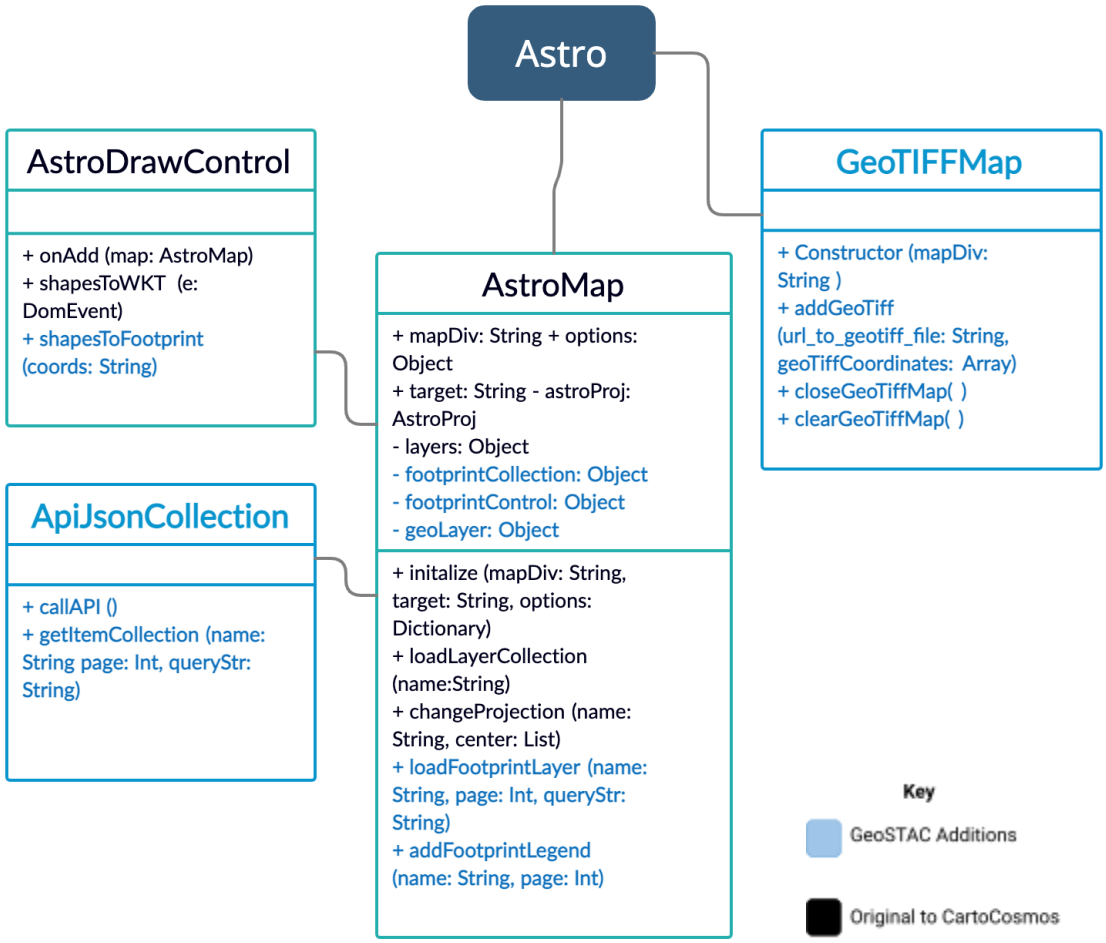

Figure 3: Astro UML Diagram

4.1 AstroMap

The existing AstroMap class is the main class in CartoCosmos. It inherits from the L.Map Leaflet class. In order to add additional layers to the Leaflet web map, having access to the L.Map class is required. AstroMap has a unique feature that separates it from the original L.Map, which is the use of a target parameter. This parameter is needed to determine which basemap to render. This target name is also needed to determine what collections to obtain within the STAC API.

4.1.1 ApiJsonCollection

ApiJsonCollection consists of two helper functions used in AstroMap for handling API fetch requests. These functions are used to return an object of JSON results or STAC items associated with the selected target body.

- **callAPI( )** callAPI returns the initial fetch of the Astrogeology STAC collections. This function is called by getItemCollection(). The output of the JSON object will consist of all the possible collections within the Astrogeology STAC API. For instance, currently the STAC catalog has one collection for Mars called “ctx_dtms” and multiple collections for Europa including “usgs_controlled_images_voy1_voy2_galileo”. As more collections are added to the STAC API, this function will dynamically collect the information in this function.

  Returns:
  - (Object) - results of stac.astrogeology.usgs.gov/api/collections

- **getItemCollection( )** This function calls callAPI() and waits for the fetch request to complete. Once completed, the function uses the JSON object to find all collections pertaining to the inputted target name. Since there may be multiple collections for one target, the function will store the associated link of the collection into an array. getItemCollection also takes in the current selected page number. When the links are obtained, the page query parameter is included in the link based on the inputted page number when getItemCollection was called. getItemCollection then fetches each link in the array and maps all results into a JSON object of individual items.

  Parameters:
  - name (String) - The name of the selected target on the Leaflet web map.
  - page (Int) - The selected page number of pagination.
  - queryStr (String) - String of the query parameters.

Returns:

- (Object) - results of the mapping of all collection items associated with the selected target.

4.1.2 loadFootprintLayer and addFootprintLegend

loadFootprintLayer and addFootprintLegend are two functions added within the existing AstroMap class. In order to have access to the map's target instance variable as well as add layers, these functions needed to reside within the AstroMap class.

- **loadFootprintLayer( )** loadFootprintLayer is called by the initialize constructor function of AstroMap when a target is selected in the GUI. This function takes in the target name as well as the page number of pagination. When initialized, this page number is always once. This function calls the getItemCollection from ApiJsonCollection and waits for the results to be loaded. Once loaded, it iterates through each item in the object and adds it to a geoJSON layer. Once all layers are added to the layer, a control is added to the map's GUI. This allows the user to toggle the footprint layer on and off. Then this function calls addFootprintLegend to add a layer legend to the maps GUI.

  Parameters:

  - name (String) - The name of the selected target on the Leaflet web map.
  - page (Int) - The selected page number of pagination.
  - queryStr (String) - String of the query parameters.

- **addFootprintLegend( )** addFootprintLegend adds a legend for the geoJSON layer of the map. This function utilizes the "leaflet-html-legend" plugin to accomplish this. This legend also allows for the option to include html segments. This function adds in a pagination component within the html segment. This will allow the user to sift through pages of footprints. In order to detect a user clicking the html components, two internal event handlers detect which buttons are selected. If these buttons are selected, addFoorprintLegend will call loadFoorprintLayer with the new page number to render the new selection of items.

  Parameters:

  - name (String) - The name of the selected target on the Leaflet web map.
  - page (Int) - The selected page number of pagination.

4.2 GeoTIFFViewer

The GeoTiffViewer class is the class that contains all the functions for displaying a given geoTiff thumbnail from a footprints assets onto a simple viewer. This class will need to be called inside of the main AstroMap driver class, in order to have access to one instance of the GeoTiffViewer. The GeoTiffViewer class consists of three functions, a constructor function, displayGeoTiff function, and toggleViewer function. These three functions will take care of all the necessary functionality for the GeoTiffViewer inside of the CartoCosmos app.

- **Constructor( )** The constructor function for the class GeoTiffViewer takes in the div id that the viewer will reside in. An empty array called imageArrays will be created as a global variable to have multiple assets being able to toggle between them. The function uses the passed div name to initialize the viewer and create the array that will track the download link of the displayed geoTiff thumbnail.
  Parameters:
  - imageDiv (String) - The name of the div ID where the Leaflet map will be displayed.

- **displayGeoTiff( )** Takes in a url for the assets file and changes the contents of the div to display the asset to the given user. This function fetches the assets from the USGS STAC server and creates a new image asset with a set size inside of the imageDiv. The thumbnail is then displayed next to the CartoCosmos leaflet map.

  Parameters:
  - imageURL (String) - The url link that contains the asset to be displayed to the user

- **toggleViewer( )** this function is used to close and open the GeoTiffViewer that contains the thumbnails, the viewer will be cleared of all thumbnails when it is closed.

4.3 GUI

A large part of the GeoSTAC project requires interaction between the user and the interface. This includes having click footprints, visualize metadata, and sort/filter this data. This means implementing additional React components into the existing GUI structure of CartoCosmos. Figure 4 provides this visualization of the added components to the GUI. These added components are highlighted in blue.

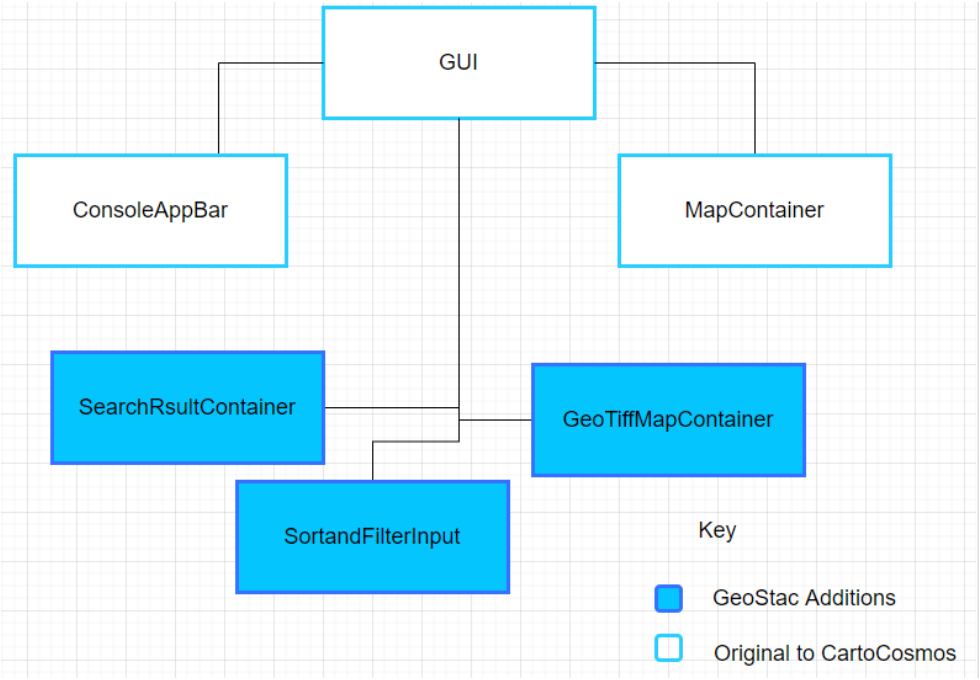

Figure 4: GUI UML Diagram

4.3.1 SortAndFilterInput

The Sort and Filter component is a sidebar that allows the user to narrow down the results according to their specified input values. It is written in HTML and CSS, with components from Google's Material UI. At the top is a dropdown list that allows the user to set a method of sorting, and sort in ascending or descending order. Underneath is a series of filters, which allow the user to narrow their STAC footprint results by keyword and/or date range, as well as spatially limit the search to an area drawn in the map section of the application. Each field has a unique HTML id so other parts of the application can access its values.

4.3.2 SearchResultContainer

The SearchResultContainer component is the sidebar that is built on top of the existing CartoCosmos Leaflet to display the metadata from a geoJson when a footprint is clicked on the map. The SearchResultContainer will also contain a button allowing the user to open up the cloud optimized geoTIFF map and add the associated COG to the map.

4.3.3 GeoTiffContainer

The GeoTiffContainer component is the base element for the GeoTiff asset viewer. It imports functionality from GeoTIFFAppBar and other modules to display the asset viewer inside of the div tag with ‘id = “geoTiffImageDiv”’, inside of the HTML page.

4.4 AstroDrawControl

AstroDrawControl is another existing module of CartoCosmos. This class adds drawing controls to the Leaflet map. The drawing tools are what is used to select multiple footprints on a Leaflet layer. In order to utilize the drawing tools for footprint searching, an additional function was added to the AstroDrawControl class.

4.4.1 shapesToFootprint

The function shapesToFootprint was added within the AstroDrawControl class in order to have access to the maps drawing controls. This will allow the various drawing tools that currently exist to be used as a selection filter for footprints on the map. This function utilizes the existing function shapesToWKT in order to retrieve the coordinates of the drawn shapes perimeter. Once the coordinates are retrieved they are passed in to a queryString which is used to query the API selection and filter down the results to features that reside or intersect within the drawn area. This function calls loadFootprintLayer from AstroMap in order to do so.

shapesToFootprint( )
Parameters:

- coords (String)- A string containing the coordinates of the given shapes perimeter.

5 Implementation Plan

In order to develop the four different modules explained, the development plan has been split into three different sections: Displaying footprints on AstroMap, collecting metadata, sort and querying searching the API and lastly displaying cloud optimized geoTIFFs next to the existing CartoCosmo’s Leaflet map. The development process has been divided into three sections as detailed in the Gantt Chart below (fig. 6): planning, development, and testing phases.

The planning phase has been mostly completed and now the development phase has become the main focus. The development phase can be divided into multiple different benchmarks to stay on track for development. The development stage started with building on top of the existing AstroMap to fetch data from the STAC catalog and

display the footprints on CartoCosmo’s Leaflet map and then add a process of pagination for the footprints in Leaflet. Creating an asset viewer for displaying cloud optimized geoTIFFs thumbnails in the app next to the CartoCosmo’s Leaflet, a filter and query search functionality, and lastly abstracting, displaying footprints metadata to the user and lastly polishing up the GUI for the new app. These sections are all being developed at the same time because the different functionalities intersect with each other. In order to display COGs and metadata, the corresponding footprints on the Leaflet map are needed.

During the development process, every class must be documented as it is implemented. Documentation is a key part to the GeoSTAC project in order to deliver a fully developed app to USGS. Documenting during the development process will allow for the most accurate representation of each class and function.

The end goal of the development phase is to create an alpha demo application. The alpha demo will continue to look like CartoCosmo's application, but with some new GUI and abilities for the STAC integration. The different benchmarks can be split into 4 different sections with a main person in charge of the section. Figure 5 specifies how the sections have been broken up.

Footprint and Pagination	**Cloud Optimized GeoTIFF Viewer**	**Fetching and Displaying MetaData**	**Filter and Search Functionality**
Amy Stamile	Zachary Kaufman	Gavin Nelson	Jacob Cain

Figure 5: Distribution of Labor

After completion of the development process, the testing phase will begin. Since the application will be used by researchers and academic professionals, the testing phase will be split into two different phases: integrated testing and user testing. The Integrated testing will ensure that the new functions and class are working and interacting with each other properly. The integrated testing will test the output of certain functionality to ensure the correct things are displayed and given to the user. The client side testing will ensure that the new GUI is easy to use and is what is needed by researchers and academic professionals. The client side testing will be conducted by people that will be mainly using the app at USGS to ensure that the GUI is up to the standards they require for their research.

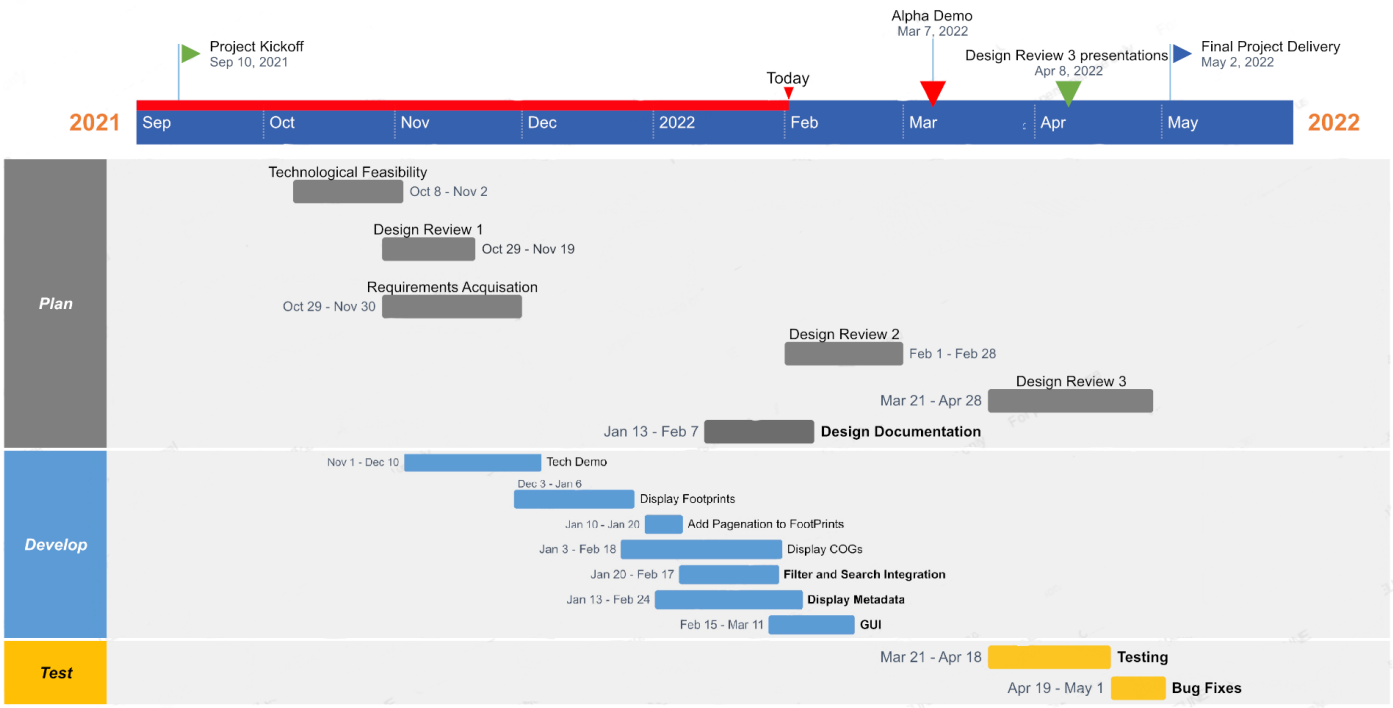

Figure 6: Gantt chart

6 Conclusion

As the need for accurate planetary data and images increases for the scientific and academic community, it is more important than ever that there is software to easily access this data. The USGS Astrogeology Science Center in Flagstaff Arizona provides the international planetary science community with new knowledge of our solar system. Without this data, new planetary exploration missions would be nearly impossible. The USGS Astrogeology Science Center needs a way for the planetary science community to interact with the analysis ready data from their STAC API.

The module architectures provided above provide a design a product that will ensure that interactivity of analysis ready data within CartoCosmos Leaflet web map. The architecture is split into two main modules that already exist in CartoCosmos: Astro and GUI. Astro is the structured back-end javascript code that includes the AstroMap class and the AstroDrawControl class. These two classes have added functions to enhance the functionality of the existing CartoCosmos web map. There is also an added GeoTIFFMap class to provide visualization of geoTIFF images to the user. The GUI is composed of React components that provide a front-end interface. Sorting and filter

tools are added to the existing GUI module to provide interactivity for the user to search footprints for a particular planetary target. As well as components to visualize the metadata associated with each footprint.

The implementation plan in this document outlines how the design will be broken up in smaller tasks. These tasks are assigned and scheduled for each member of GeoSTAC. This plan will set up the team with success in providing USGS with an enhanced web map that provides the user the ability to interact with analysis ready data.