Glacies Indicium

Requirements Specification

Version 3

11/16/2020

Project Sponsors	**Team Members**	**Faculty Mentor**
Dr. Heather Lynch	Beck Bohnker	Andrew Abraham
Dr. Mark Salvatore	Joe Carter	
Helen Eifert	Kassandra Coxen	
Brian Szutu	Logan Garrett	
	Zach Spielberger	

Client Signature: _____ Date: 11/20/2020

Client Signature: _____ Date: 11/20/20

Team Signature: _____ Date: 11/20/2020

1 Introduction

Antarctica is one of the last great unexplored wildernesses. It is the coldest place on Earth, larger by area than both Europe and Australia, and bordered on all sides by at least a thousand kilometers of icy Southern Ocean. Because of this, Antarctica is extremely inhospitable. Study of this frigid desert continent has the potential to significantly expand our understanding of the scientific and geological processes that govern the formation of rocky planets on Earth, but those who seek to explore it are constrained by its inaccessibility. The development of satellite imagery, however, has greatly facilitated our ability to study this harsh and remote environment.

We are Team Glacies Indicium, and our clients are Dr Mark Salvatore and Dr Heather Lynch who are from Northern Arizona University and Stony Brook University respectively. They are planetary and ecological scientists who are trying to learn more about Antarctic geology and currently have a dataset of more than 60,000 high-resolution satellite images of Antarctica, which they want to use to document and study Antarctica's physical properties and surface composition. They have hired our team to build a web-based visualization tool, which will greatly reduce the difficulty of studying and interacting with large amounts of the data. The rationale for our product revolves around the fact that when this information is easy to obtain and interact with, the potential for new discoveries increases. Our product will be geared towards members of the scientific community that may not have a lot of computer experience, so we in turn want to deliver a product that allows scientists access to this satellite data in a way that is as fast, easy, and intuitive as possible.

2 Problem Statement

Our clients search for new insights into Antarctic geology by using Graphical Information Systems (GIS) software to examine satellite images specifically in the form of a raster image. For example, a normal image on a computer is made up of three bands, a red band, a green band, and a blue band, that are added to produce the full range of colors; hence, the result is an RGB digital image. A raster image, though, isn't limited to a specific number of bands and can also map these bands to places on the earth using geographical coordinates.

In some cases, the raw images our clients work with only have eight bands representing different ranges of the electromagnetic spectrum, but some images may have more. This is because, as humans, we only see a small range of light in this spectrum, but we can still collect data from other ranges of the spectrum. Our clients have done mathematical modeling and analysis with this raw spectral data to compute and derive new data bands representing landscape data, such as presence of snow, shadows, and ice, and even identifying certain rock types. Using GIS software, like Google Earth and ArcGIS, they can display these different wavelengths or landscape data and map them to their location on Earth. Using different combinations of the spectral wavelengths, they can visualize what it would look like to the human eye, or they can view single or multiple bands of landscape data.

This analysis is valuable to the community, as well as to the world, because it helps build our understanding of the Earth's least understood continent, Antarctica, and its role in the world's earth systems. In some cases, it even showcases how Antarctica is not being as affected by climate change as other regions in the world. Currently, they examine these multi-layered satellite images layer-by-layer, one image at a time, which can individually visualize about five square kilometers. The first problem is that it's easy for a scientist to examine one image, which in size can be about two GBs, but it's impossible to interact with the data on the continental level as it can take up multiple petabytes. Ideally, a geologist would want to visualize a map of a larger region, as this would give them greater insights to drive future field expeditions as well as a new perspective of the data on the continental level. Furthermore, the main challenge is that geologists don't have a means to query the entire dataset by features such as geological rock type. If they did it would allow researchers to skip the remote sensing steps and move straight to the geological analysis and further broaden the type of scientists who can work with satellite imagery therefore removing the requirement for experience with remote sensing.

3 Solution Vision

This is where our team comes in: we have been hired to build a web application that allows scientists everywhere to explore this collection of satellite data, by visualizing the raster data in an interactive web map. The tool will allow users to individually toggle as many as three layers of data in each satellite image, providing access to a wide range of information including wavelength reflectance, rock type, and presence of snow, ice, or shadow. The tool aims to greatly improve the user's workflow efficiency and ease of access to the geological data by implementing a Graphical User Interface (GUI) that equips the

user to view and filter the data based on geographical region and various geological attributes. Finally, the most important feature our product will provide is the ability to query by actual geology, which the current technologies our clients work with do not satisfy. The belief is that such an application will facilitate the global scientific study of Antarctica by making this vast trove of data interactive and easily accessible to scientists working in institutions across the globe.

This web application must be able to quickly retrieve and display a large number of raster images at any given time. Python will be used to implement this project along with specific libraries like Terracotta, which provides a framework for raster images to be stored in a database and returned to the user as PNGs, single band RGB- or greyscale-based raster images, to display onto the map. Our current solution vision incorporates this process and is shown below in Figure 1.

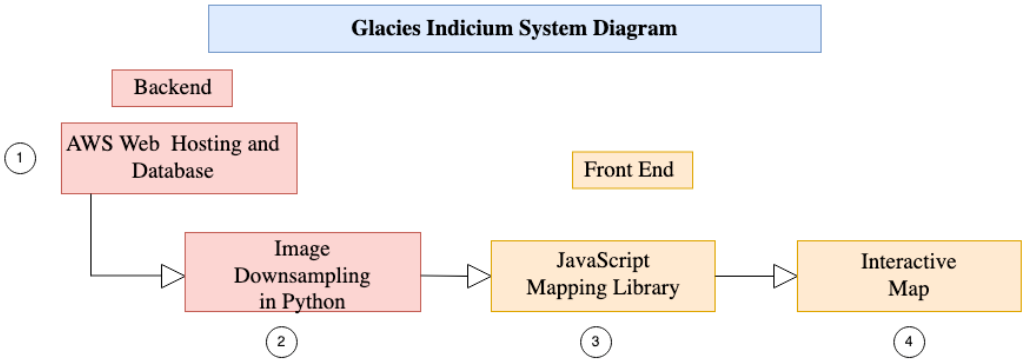

Figure 1: Envisioned System Design Diagram

The current design is divided into four parts that are each components of either the front or back end of the web application. The backend will include the website and database hosting along with down sampling the raster images to provide faster loading times for the user. The frontend takes the information provided from the backend and displays it onto an interactive map by using a JavaScript mapping library.

1) Amazon Web Services Website and Database Hosting – Amazon Web Services will be the primary host for the website along with the database where the raster images will be stored. AWS provides a reliable hosting services that allows for a large amount of data storage and will also provide a fast retrieval time for displaying raster images onto the map.

2) Image Downsampling – Terracotta automatically provides functionality for downsampling the raster images, which converts them into smaller images to improve the overall performance of the application – a crucial component to ensure our solution remains feasible when the user wants to view a large amount of data.

3) JavaScript Mapping Library – Once the files are downsampled, Terracotta uses Leaflet, which is a JavaScript mapping library, to quickly tile the raster images and display them onto an interactive map for the user to navigate the data.

4) Interactive Map – Leaflet provides a framework for many of the key functionalities needed for the user to interact with the map. This includes zooming in and out, moving the map, and querying for specific information from the database.

4 Domain Requirements

There are several key requirements our envisioned solution must fulfill. These requirements contribute to every aspect of our application: from usability to large scale data storage to requirements derived from working with raster images.

Figure 2: Example of Graphical User Interface (GUI) from http://www.penguinmap.com/mapppd

The first major milestone is finding an effective way to store all the raster images, this is more than just using AWS Website and Data Hosting because of how much information is in just one full raster image. The application also needs the ability to find different raster images in our database, so images can be displayed to the user in a timely fashion. The next requirement involves employing a Graphical User Interface (GUI) that is straightforward for first-time users to understand how to interact with our application, like that of the example in Figure 2. Then, for users that wants to interact with different sections of the map, there needs to be a way that lets them filter and search for specific data anywhere on the map. The last set of domain level requirements involve making the data more accessible to the user by providing functionality to

export the data for use in external applications. In the next several sections, we will explain how we plan to meet each of these requirements.

For the sake of clarity, for the remainder of this document specific requirements that this team would like our client to acknowledge and sign off on will be **bolded**.

5 Functional Requirements

The functional requirements for this project will be separated into two different parts. The first is the web application's interface, which includes displaying the raster images and the map application's GUI capabilities. The second part is retrieving these raster images from the backend using a database server, and filtering the information to display to the user.

5.1 Graphical User Interface(GUI) and User Functionality

The graphical user interface for this project is an interactive web application for displaying a map of Antarctica that visualizes raster images. This application's interface must have three key requirements: the map display populated with raster images, as mentioned, plus the ability to navigate the map including zooming in and out, and the ability to query the map for specific information by spectral band, date or location.

The most important requirement of this application's interface is **displaying a map** that can quickly display multiple raster images over Antarctica at one time. In order to provide a web based mapping application with as much functionality as possible to the user, Terracotta along with Leaflet will be primarily used. Terracotta is a Python tile server used for quickly displaying raster files stored on a database. To display these files, Terracotta uses Leaflet which is a JavaScript library for displaying dynamic and interactive maps both on mobile and desktop web browsers. An example of an empty Leaflet map is shown below in Figure 3. The map for our application, however, will only display the continent of Antarctica.

This application must also provide options for the user to **zoom in or out** in the map. The options to zoom in and out will be presented to the user using two buttons in the top of the map as shown above. The user will also be able to adjust the zoom level by scrolling with their mouse wheel.

Figure 3: Example map built using Leaflet. This map shows a base map before any data has been added.

Users must also be able to quickly **search for a specific region** within Antarctica and then the application must **filter the data** and return the results. To accomplish this, a dropdown will be presented to filter by specific values, such as region or date. Users must also be able to type a specific search query such as "10-27-2018" or "dolorite" into an input field in order to locate their desired information. When the information is loaded into the SQL server provided by Terracotta, keys will also be given to properly label the information when it is presented on the interface. Keys are the keywords used to filter the data by band, region, or other attribute that the user specifies. The application below in Figure 4 was created using Terracotta, and in it's current implementation, can display one raster image onto the map. The available keys for this application are date, tile, and band, and each one has their own input field as well. For this project, the possible keys the user can access will be rock type, the date of capture of the raster images, and location. For any given key, this project must be able to query the database and return any matching results back to the user interface.

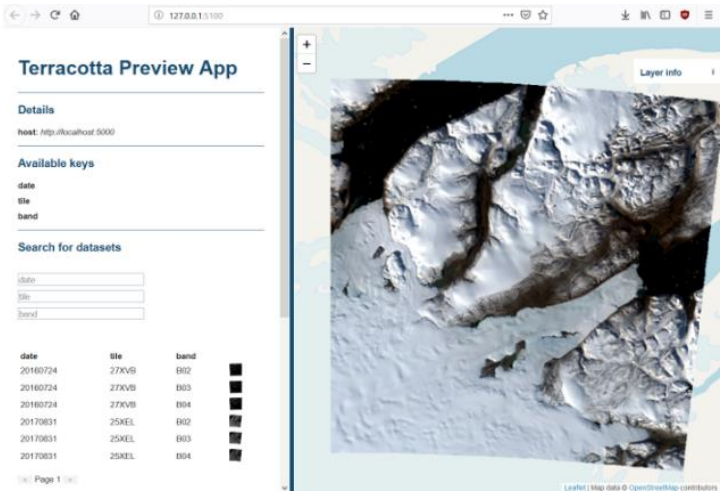

Figure 4: Example Terracotta Application

The above application in Figure 4 also shows the combined requirements of this project's user interface. This application allows for users to zoom into the map by using the '+' and '-', buttons in the top left of the map or search for specific data in the toolbar on the left side of the map.

5.2 Filtering Data by Bands & Serving it to the User

The source raster data provided by our client will be in GeoTIFF format, a raster image format containing geographical location data, as well as anywhere from 8 to 18 bands, with the number of bands depending on the availability of various geological data information throughout Antarctic regions. One difficulty in working with multi-band GeoTIFFs is that Terracotta requires that each band be ingested as a single image file. Because of this constraint, we will need to perform a preprocessing step on the source data to **split each source image** into separate images for each band, with the bands keyed by filename. This step will only need to be performed once prior to having Terracotta construct its database. Once this is completed, the front end of our web application will be able to filter the available images by using band numbers as keys, in addition to the other keys we specify in our file naming convention.

The multi-banded GeoTIFF image format has been in use for a long time, and there already exists a special sub-format called Cloud Optimized GeoTIFF (COG), which **restructures the data** within a GeoTIFF image so that subsections of the image can be accessed using HTTP Range queries. COG conversion of the source data will only need to be done once, so this will just be one more preprocessing step to construct the database.

The source of data for our application is a collection of hundreds of gigabytes of multi-layered images, and each image contains several gigabytes of information. Because our web-based application needs to be reasonably fast and responsive by loading in under a minute, serving these full images directly over the internet is obviously unfeasible. Our solution to this problem utilizes the Terracotta open-source Python tiling image server. Terracotta works by performing a preliminary ingestion of all our source data into **an SQL database**. Once the database has been created, Terracotta **produces small PNG image tiles** from database queries as a client requests them, and then serves these to the front end for display as layers on a Leaflet map. By incorporating Terracotta this application will be able to serve lots of small PNG images, each roughly several kilobytes in size, which are constructed from only the data the web client needs at any given time. This overcomes the challenge presented by reducing the size of the data to deliver users responsive and interactive access to such a large volume of source images.

6 Performance Requirements

This application, if scaled to the whole continent of Antarctica, could potentially be working with several petabytes of data (~1000 GBs). As we have discussed previously in our Technological Feasibility report, this is an extreme technical challenge, as it is not feasible to serve this scale of data across the internet in seconds. Despite this, our tool still needs to remain useful to our clients as well as scientists surveying Antarctica. To deliver an application that solves this overarching requirement, we need to serve data in a timely fashion. Our current methodology to accomplish this is to serve smaller "downsampled" versions of the data when the scale of the map is larger, and serve more accurate "upsampled" versions of the data when looking at a more 'zoomed in' region of the map. Our minimum viable product will be capable of surveying a select few areas in Antarctica including the Dry Valleys, as shown below in Figure 5, which is a regional hotspot for geological study. We estimate the data for this area will consist of around 100 raster images that are about 2 Gigabytes (GB) each in size. The following sections outline performance requirements related to potential load times alternatives, our plan to utilize SQL to serve the data across the internet quickly, and some statistics for how big we expect the databases to get. Initially, we plan to **map the Dry Valleys**, before transitioning to other areas of the continent.

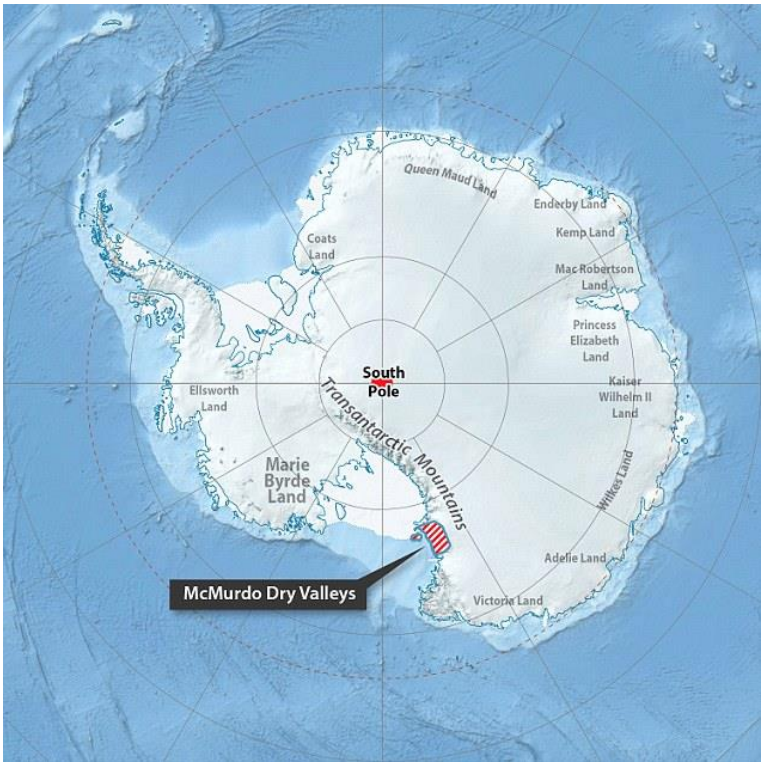

Figure 5: Map of Antarctica referencing McMurdo Dry Valleys

6.1 Load time

In our tests with Terracotta, it took about 30 seconds to begin rendering artifacts on the map from the users perspective after generating a database with a mosaic of a single band across eight images (~1GB total). This is due to the program using the SQL database it generated from the raster images to create PNGs to send to the user for the area they are looking at in question. This is the alternative we are currently utilizing to avoid sending full size raster images across the web.

In our prior discussions, Dr. Salvatore mentioned that it would mainly be a convenience for an area to load in around 15 seconds, but that it may not be possible due to the whole region being made up of well over ten times the number of images than our original test. In addition, the images we previously sampled were also much smaller in size compared to what we're estimating a typical raster image in the Dry Valleys will be. Considering that the data we would be dealing with would be around 100 times bigger, we've decided that an initial load from the user's perspective should take, at minimum, **one minute to load**. The Dry Valleys themselves constitute over 4000 km^2 so a minute load time does seem reasonable, but the valleys themselves only make up about 0.03% of the continent. This encourages us to want to load the

Figure 6: Map of Antarctica referencing McMurdo Dry Valleys

region faster than the previously mentioned requirement, in case we want to expand the areas included on the map as a stretch goal, while still retaining product usability. The more important requirement than load time, however, is that we should display a **progress bar** to elicit that work is being done, especially if we're unable to meet the minute requirement.

In the above test case with Terracotta, once the initial rendering of the image was completed panning around the image was relatively responsive. Images are tiled across the user's screen representing the different data of a requested band in roughly half second intervals. Once again, when zooming in or out, images will be retiled across the screen to represent different resolutions. This means at a high zoom level the images will be a lower resolution than if you were zoomed in. To have a responsive tiling process (and hopefully retain the **half-second tiling time**), we are planning on making changes to the way we are generating databases, which is described below.

6.2 Handling databases

To retain usability, our plan is to downsample the dataset by multiple factors and at minimum have **two sets of raster data** represented in SQL: a high-resolution dataset and a low-resolution dataset. We've decided that we may generate multiple versions of the previously described SQL database that our

program uses in order to generate different resolution PNGs with the requested data from a particular band for a certain region. We hope that this will help combat load times in the sense that when looking at a relatively large area on the map, since the user doesn't need as accurate data over a large area and could request data from the smaller, low-resolution dataset. Then, if the user wants to look at a smaller area, they will begin requesting access to the larger, high-resolution dataset. This will, ideally, limit the amount of data we are sending across the web and enable a more responsive interface by only requesting the data that the user needs. If this does work out, the team may consider expanding the two sets of data to three or four to see if we can gain further speedup.

In addition, generating the SQL databases can be a costly and time-consuming endeavor, but they typically only need to be generated once. From there, users can continue to request access to the data through the web app until the databases needed to be updated as additional data is captured and must be added to the raster set. We still are attempting to be conscious of the time this takes in case the database does expand under use by the Polar Glacial Center, but since it happens so rarely, it has been recommended to our team to **not set a time requirement** on database generation.

6.3 Size and Scale

It's difficult to estimate the size of a database until it's created but we expect the size of the generated database to not be significantly larger than the size of the original dataset. We already estimated the region of the Dry Valleys to be around 100 images each about 2 GBs in size (200 GB total). If indexing the data does increase the size, we estimate the high-resolution database to be ~210 GB in size. As would be expected, the downsampled datasets would be scale factors smaller than the original 200 GBs so we are planning to **downsample by a factor of 1/10** for the low-resolution database putting it at ~21 GB in size.

7 Environmental Requirements

In addition to our project's functional requirements, there are a number of environmental requirements, set mostly by our clients, to which our project implementation must adhere in order to align with the final product our clients have in mind.

7.1 Working with raster data

One major requirement and challenge is that our product must work with our client's existing satellite data, which is currently available to us as raster data, specifically **GeoTIFF files**. This requirement does not

pose an obstacle for us since there are many solutions available for working with raster data. Additionally, raster files are easily able to be converted to other formats using the translator library GDAL (Geospatial Data Abstraction Library), if necessary. The main obstacles that arise concern the properties of raster files, such as file size for example, which were previously discussed in this document.

7.2 Web Application Interface & Exporting Data

Our client has asked that the map and interface be a **web application** so that it may be more easily accessible to other scientists in their field. However, our clients have also mentioned that they may need to access data in remote areas of Antarctica that have limited to no connection to the Internet. Our solution for this is to include the ability to **export "snapshots"** or views of the map that include filters for queried values and selected areas that the user has specified. Also, since the typical user is already familiar with raster images they can **request a zip** containing the raster images for a particular area.

7.3 Using Python to process data

Because our client and their team are already familiar with and currently using Python for other analysis tasks, it would benefit them to **use Python** for the product we are creating as well. This allows them to more easily maintain the system. Additionally, our clients have discussed that Python and the respective geospatial libraries are considered more widely used and understood by peers in their fields. This would allow our solution to have a wider impact amongst this community. This requirement is more of a client preference, as they have also suggested R would be acceptable, as well as any other viable software solutions. Because of the simplicity, extensive documentation, and standard use of Python for processing geospatial data, there is more support for this solution, so we have agreed to stick with the client's preference.

8 Potential Risks / Challenges

The implementation of this geological data tool does not involve many real risks. There are, however, a few challenges which we need to get right to be confident that the tool is scientifically useful. These are:

- Ensuring compatibility between the projection of the source data and the map onto which we project the data
- Maintaining the integrity of the data through the downsampling process

- Ensuring that the source data keying scheme is consistent, accurate, and relevant
- Making imagery and location information easy to export for reference offline

Ensuring projection compatibility is crucial to our application. We need to make sure before this tool is deployed that the interactive Leaflet map we are using is in the same projection as the source data we are overlaying onto it. This is important because any kind of projection mismatch will result in imagery being incorrectly located on the map, which would eliminate any possibility of scientific study because the tool would not align with reality.

When we downsample the source imagery, we make a compromise by trading data resolution for transfer speed. We need to be very sure, however, that our procedure for resampling the source imagery does not affect the scientific integrity of the data in any way. We will ensure this by making direct comparisons to unaltered and resampled image data and making every available effort to establish and verify that the resampled data is materially unaffected by the resampling process.

The **keying scheme** is central to how users of our application will get to the imagery they want. Because our keying scheme gets baked into the SQL database at the time the database is created, any mistakes in the keying will require that the mistake be corrected and the whole database rebuilt, which, given the sheer volume of source data, will inevitably consume valuable time if it had to be done on a full dataset.

Exportability has been one of our core requirements from the start. The only real risk here is that our tool does not do enough to ensure that all the information one would need to use exported data offline is present and accessible in the final export. We do not want someone to try to export some useful imagery and print it out, only to later realize that the export does not contain everything they wanted it to.

9 Project Plan

The plan for this project is currently broken into two parts: the team website and the final application. Moving forward, our team will be focusing on utilizing Terracotta with Leaflet to deliver an alpha prototype by the end of the second semester, along with a completed team website outlining our team and the project goals. In the second semester, more testing on the prototype will be conducted followed by implementing changes based on the client's review of the early system. The complete team schedule can be found below in Figure 6. The major goals the team will be achieving are the following:

I. **Completing the team website:** This will be completed November 16th, 2020 and will include all the finalized documents from the first semester including the team inventory, team standards and technology feasibility documents. The website will also include multiple pages with simple navigation to each page within the website.

II. **Creating an early prototype with working requirements:** An early demo of the project will be presented on November 9th, 2020 and will be run locally with a smaller set of raster images that will grow as the project becomes further implemented.

III. **Designing and hosting the database on the AWS backend:** During second semester, we will take the first step in implementing the final version of this project, which will be designing the AWS database backend to host the raster images. By hosting the files outside of a local machine, it will allow for anyone in the world to access and view this application.

IV. **Implementing the interface with search functionality that connects to the AWS backend:** After the backend is implemented, more testing will be done to ensure that the front end will properly connect to AWS that will allow users to search and filter the information.

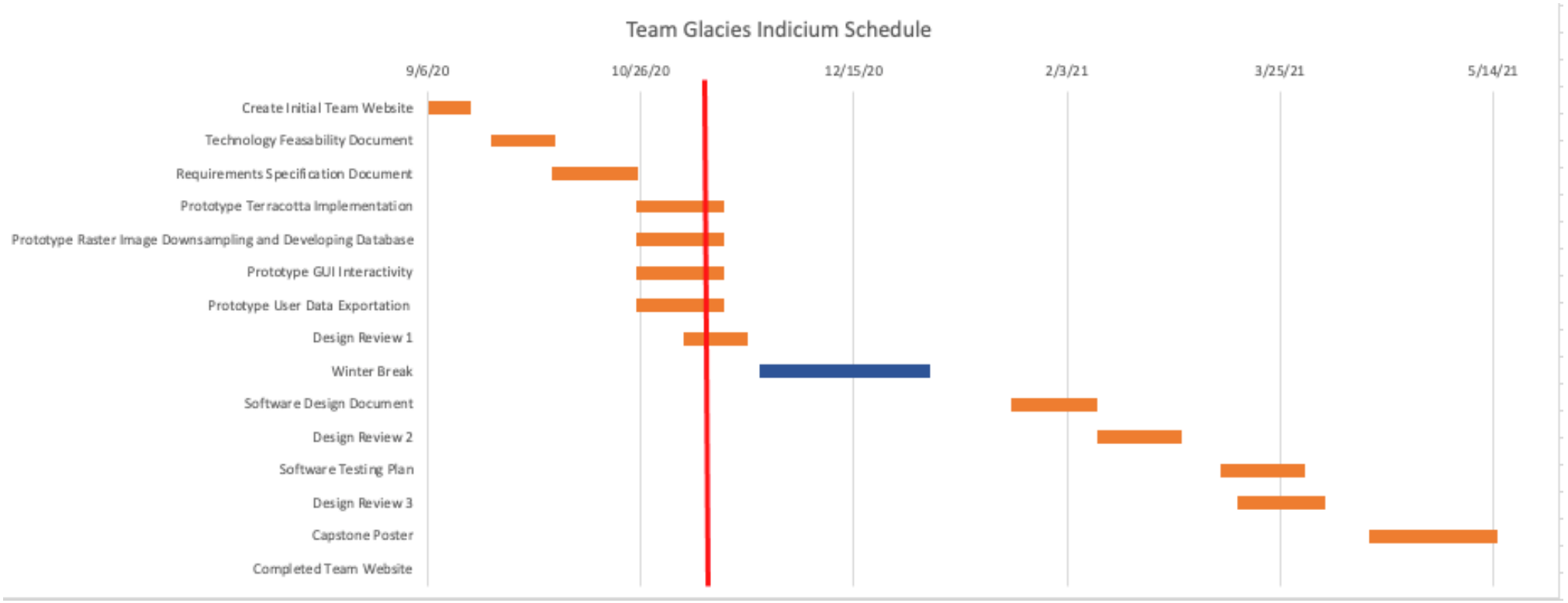

Figure 7: Current Team Schedule

10 Conclusion

The hostile Antarctic environment presents a unique set of challenges to scientists trying to understand it. We aim to build an application that helps scientists and explorers in organizations around the world unravel the mysteries of this largely undiscovered frontier, expanding not only our understanding of our own planet, but also our knowledge of other planets that seem potentially inhospitable. Overall, the benefit of our product is that it provides ease of access to the large amount of data used to map Antarctica, expanding the catalog of available area for research and further enabling scientists to gain a better understanding of the region.

Our application will present the scientific community with an interactive map of Antarctica, and an interface through which researchers can view and analyze a large and diverse set of raw and derived satellite imagery. The bands of data for each image will be viewable either as single bands in a binary color scale or interpolated together as RGB channels for geological differentiation. Imagery and location data will also be easily exportable for use in offline contexts.

The purpose of this document has been to establish the domain-level, functional, and non-functional requirements of the project, to provide our team and our clients with a roadmap of specific features that we expect the final application to include. By implementing our application according to the requirements set forth herein, we will ensure that the goals we set for ourselves as a team are aligned with the goals of our client and that achieving these goals will make everyone happy at the conclusion of the project. Our preliminary research into the technologies we will use to accomplish these goals has yielded some very positive progress toward the development of a functional prototype, and we anticipate going forward that the project will culminate in a resounding success.