Glacies Indicium

Final Report

Version 1
04/19/2021

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	

Contents

1 Introduction

Antarctica is one of the last great unexplored wildernesses. It is the coldest place on Earth, larger in area than both Europe and Australia, and bordered on all sides by at least a thousand kilometers of icy Southern Ocean. The study of this icy desert has the potential to significantly expand our understanding of the Earth's systems, but the environmental conditions pertaining to the harsh climate cause Antarctica to remain extremely inaccessible. However, as satellite imagery technology has developed, it has facilitated our ability to study this tough and remote environment.

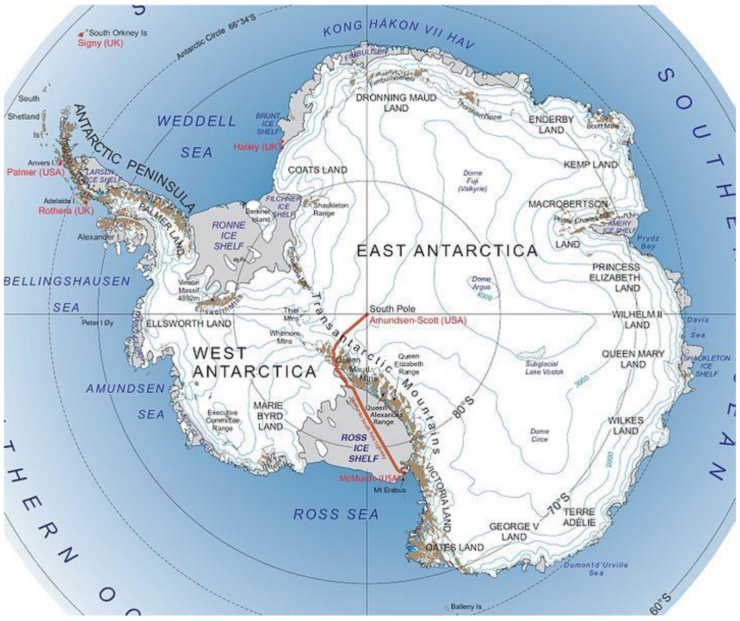


Figure 1: Full extent of Antarctica

Our clients are Dr. Mark Salvatore and Dr. Heather Lynch who are planetary and geological researchers from Northern Arizona University and Stony Brook University, respectively. They are working towards analyzing 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 such a large amount of data. Our product was created with the intent that members of the scientific community may not have a lot of computer experience, so we have aimed to deliver a product that allows scientists to access this satellite data in a way that is simple, intuitive, and fast.

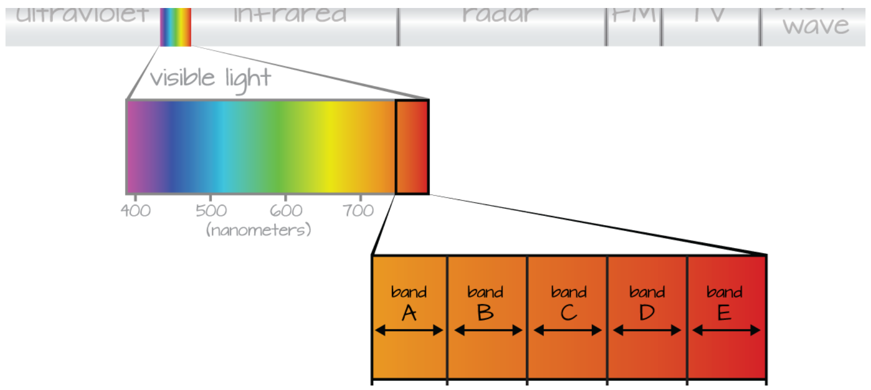


Figure 2: Range of light visible to the human eye, which can be separated into bands and stored in raster images. Values in raster images can also come from outside of the relatively small range of human-visible light.

The research our clients are conducting involves searching 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, also called a GeoTIFF or TIFF, after the file extension used for the images: .tiff. To illustrate what a raster image is, consider that an ordinary digital image on your computer is just a composition of three layers or bands: a red band, green band, and blue band. These three bands are added together to produce the full range of colors that we see. As illustrated in Figure 2 above, a raster image, is not limited to a specific number of bands and can contain data for wavelengths of light that are outside of our visible range. Raster images can also include bands created from processing the raw spectral values using mathematical formulas, resulting in derived bands that represent geological qualities, such as presence of ice or certain rock types. Furthermore, a special type of raster images, known as GeoTIFFs, can contain metadata that includes the coordinates of the geolocation for that image. Analysis of these raster images allows researchers to draw scientific conclusions about the environment and geological composition of Antarctica.

Currently, our clients' research and analysis process are hindered by technological barriers, which our product aims to resolve. They are able to view only a single image and a single band at a time. Not only does it take longer to view the data this way, but it would be much more intuitive to view the data on a larger scale to quickly understand where individual images are from and to easily identify geographical relationships between them.

Because some of the most extreme weather conditions on Earth take place in Antarctica, research of this continent allows us to formulate scientific projections about the geological and environmental phenomenon occurring in other areas of the world as well. Scientific advancement in some of the world's most important areas of study like climate change, energy, and resource consumption, are made possible by contributions from planetary and ecological research of Antarctica. Thus, a visualization tool for the large amount of satellite data for this continent will make this data more accessible, promote study of the area, and benefit not only the scientific community, but the entire world.

2 Process Overview

Our team started our project with the understanding that everyone staying up to date on important developments was deeply important to our success. Our plan to accomplish this included communicating with our clients, mentor, and other members of our team at least once a week. Each semester we scheduled a weekly meeting with our clients and another with our mentor. We also had either one or two meetings a week with just our team, this way all updates were common knowledge for everyone working on the project. When there was something too important to wait for the next meeting, our team had a Discord server that could be used to communicate with other members of the team. Other forms of communication with our clients and mentor including writing multiple papers documenting the progress of the project and emailing back and forth. These started with us gathering requirements and eventually lead to how we planned on testing the alpha prototype.

We needed a place to share our code base from, as well as the documents we were collaborating on, so we utilized two different software platforms to help us work together easily and virtually. Our code based was stored on GitHub, which allowed each member to have access to the codebase on their own machines. The rule we established with our GitHub repository required another member of the team confirming that a push or commit request would not break the whole project before it was merged with the main branch. Then, for each document we created, Microsoft Teams allowed us to create the PowerPoints and Word documents that were delivered throughout development. This solution made it possible for our team to collaborate on the project in real time and provided a solution that worked well for us.

We each took on different roles within the team. Joe became our team lead; it was his job to delegate work, make sure we all knew about deadlines, and to run our team meetings. Kassandra was our reporter, which meant that she oversaw keeping meeting time records. Beck was the customer communicator, so it was their job to relay all important information to our customer. They were also responsible for asking the clients for any needed data. Logan took on the role of quality assurance. His job was to manage our GitHub repository. Finally, Zach considered software implementations and compatibility as the team's architect.

3 Requirements

In the first phase of the project, we focused on understanding our clients' problem and their need for a software solution through weekly meetings. This process of requirements acquisition involved back and forth dialogue to establish common understanding of their work and end goals. In addition, we worked to develop the different types of functional and non-functional requirements of our envisioned solution, which we then outlined in the Requirements Specification Document, that was approved by our clients.

In the Requirements Specification document, we outlined functional, performance, environmental, and domain requirements that we had developed from conversations with our clients. Our team and our clients agreed that the minimum viable product would be an interactive, web-based map visualization of the satellite images that our clients would provide us with.

The first set of requirements involves manipulating the GeoTIFF data given to us by our clients to ensure compatibility with Terracotta, the framework we used to display to a webpage. This is done in our preprocessing system. This system is responsible for ingesting coordinate and regional hierarchy data. Once this information is in the system needs to take the Tiffs so it can rename and create mosaics them by region. Lastly, it takes the mosaics and breaks them down by band, so they can be ingested into Terracotta.

The next set of requirements apply to the web application. The interactive map needs to be populated with raster images and needs the ability navigate by zooming in and out. It also needs the query by specific information such as: spectral band, date, and location. Another key requirement that needed to be implemented was exporting the currently displayed raster image.

4 Architecture and Implementation

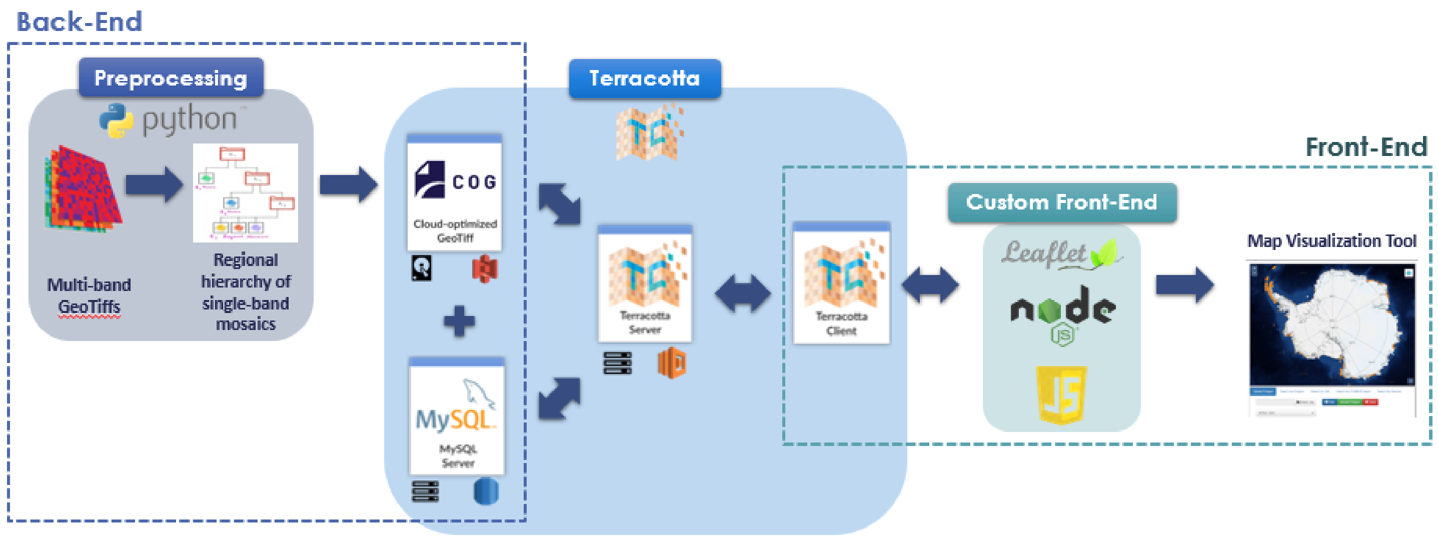


Figure 3: Architectural overview diagram to illustrate flow of data through components

4.1 System Overview

Our application was created with the goals of providing a simple and fast large-scale visualization of satellite images for geologists to use for their research and analysis. It is essential that our application can handle large amounts of data while still ensuring a quality and efficient performance for the user.

The architecture for this project is divided into two key components connected by the open-source tile server, Terracotta. The back-end component contains our preprocessing module, the GeoTIFF dataset, and the MySQL server that are generated using Terracotta's ingestion process. The front-end component utilizes JavaScript, NodeJS, and the JavaScript library, Leaflet, to produce a customized, interactive graphical user interface. As shown in Figure 3 above, these two components are connected using a Terracotta server that facilitates communication between the database and the client, so that the map visualization is populated with our data.

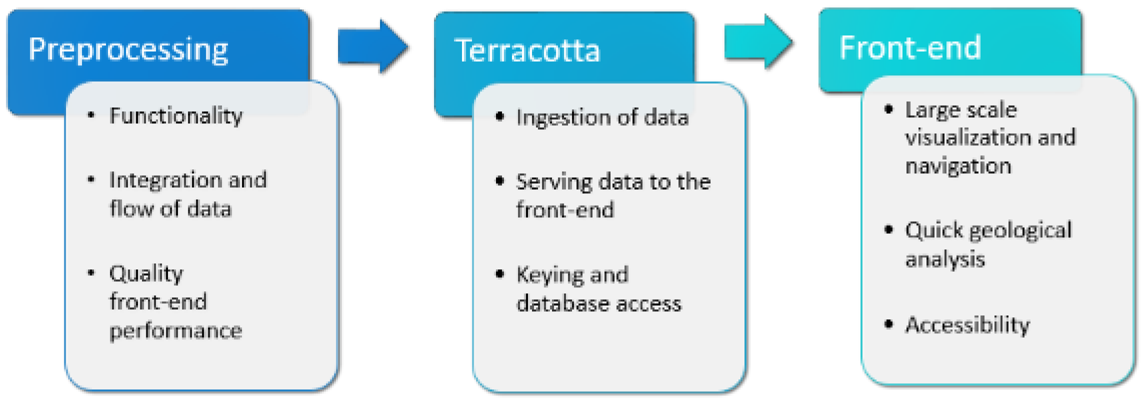

Preprocessing
Functionality
Integration and flow of data
Quality front-end performance
Terracotta
Ingestion of data
Serving data to the front-end
Keying and database access
Front-end
Large scale visualization and navigation
Quick geological analysis
Accessibility

Figure 4: Outline of purpose and goals for each component of the system

Figure 4 shown above outlines the main goals of each component within the system. Our system was built around the use of Terracotta, as it provides the framework for ingesting raster data, keying the data and producing a database, and serving it to the front-end in a format that can be quickly accessed. This would allow us to focus on the development of the other two components and guarantee our key requirements would be met.

We aimed to prioritize the preprocessing module, as one of the major challenges we anticipated was ensuring quality performance and load times within the application, knowing that our clients are working with over 1 petabyte of raster data in total. While this application would likely not be able to visualize that much data at once, we aimed to have a quality performance for at least 200 images, which can be around 400 gigabytes of data or more. Terracotta helps with ingestion and optimization of the images to ensure quick and seamless tiling within the application. This process involves correlating each raster to the region it belongs to, organizing the files into a regional hierarchy, creating a mosaic for each region, and splitting the mosaics by band to produce single-band regional GeoTIFFs that Terracotta can ingest. The preprocessing module helps to ensure integration of the components, proper data flow through the components, and functionality of front-end features that rely on proper formatting and organization of the data.

The front-end component works to quickly deliver a large-scale visualization of the satellite images that is easy to use and understand. Our product is intended for use by scientists who may not be educated in using typical geological visualization software, so the graphical user interface needed to be simple and straightforward to allow any type of researcher to quickly analyze and draw conclusions from the data. Key features of the front-end are a graphical user interface that allows the user to navigate through the data, including filtering the data by band and region. This component also works to meet our clients' requirement of providing an exporting feature to make the source data more accessible to the scientific community.

In a typical use case, our clients would run the preprocessing module on their raw data to produce the regional hierarchy of single-band mosaics. They would then be able to use Terracotta to ingest the mosaics, which optimizes the data and keys each file in a MySQL database, and start a Terracotta server. Once the data has been served and checked for errors, the product could be deployed, and anyone would be able to access the application via the web and interact with that data.

4.2 Preprocessing

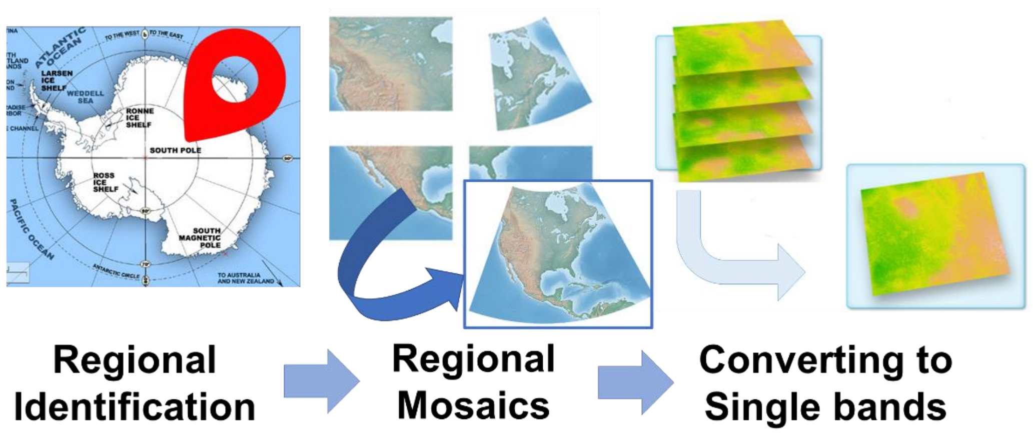


Figure 5: Steps in the Preprocessing module

The main goal of our preprocessing module is to organize and reformat the data given to us by our client to make it usable in a web-based map application. To accomplish this, each image needs to be matched to its corresponding region so that the images can be organized by region and regional mosaics can be generated. Then, the images need to be split by band to produce single-band images that Terracotta can ingest and serve to the front-end. Lastly, the images need to be renamed to allow for keying of the regions and bands in creation of the database. This entire process is illustrated below in Figure 5.

This module first identifies which region each image belongs to using a haversine formula, which finds the shortest great-circle distance between two points on Earth. For our case, the two points are the center of each image and the centers of each region, as provided to us by our clients in a CSV file that we converted to JSON called `all_regions.json`. Comparing these results, we determine the region an image based on which pair has the shortest calculated haversine distance. This allows us to then create the regional hierarchy similar to the structure in Figure 6, shown below.

Once the regional hierarchy has been created, mosaics, or compositions of multiple images, are created

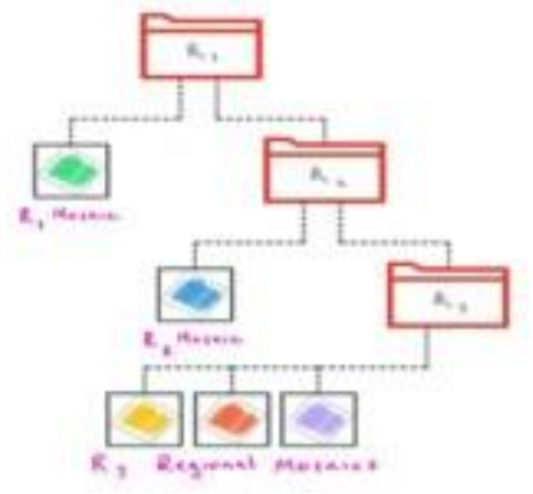


Figure 6: Regional hierarchy file organization structure

for each region. In creating the mosaics, we aimed to reduce load time and increase performance on the front-end, as there are less images to retrieve and load for each region. Then, as previously explained, the mosaics are split by band to produce single-band images, since Terracotta cannot ingest multi-band GeoTIFFs. It is during this step that our client has the option of downsampling the images should their file size be relatively large, which is a feature added to help maintain performance for their large quantity of data, should larger datasets cause poor front-end performance or long runtimes. When this process is complete, the file names are renamed to include the region and band. This allows Terracotta to key the images in the database properly to help facilitate filtering on the front-end. It also is crucial to the exporting feature that they are named according to this convention, as the front-end traces back to the source images using the file name.

4.3 Terracotta Server & Client

The Terracotta Server is the first script to be run before the user’s webpage is loaded. The server will optimize the data and create the MySQL database using band and region keys included in the filename structure. The PNG images resulting from this process will be sent to the Terracotta client to display in the interactive map. The biggest benefit of using Terracotta is that it provides the framework to serve the images to the front-end client and create a completely functional base map visualization, leaving the majority of the development on the front-end for customization of the graphical user interface.

4.4 Custom Leaflet Map Visualization

On the front-end, we added a dropdown list for the available bands and a region list with search bar to select the region of interest. The user is also able to change the colormap to help show areas of geological interest or contrasting values more clearly. Layer info is provided as well, to include information about the data being shown, including a summarized version of the metadata. Upon exporting, the images currently in view are retrieved based on the active band and region keys, and a GeoTiff of that region is provided as a download. Each of these features is accomplished through JavaScript, NodeJS, or Leaflet, and given accurately preprocessed images, these features are standard in web applications, so their implementation is straightforward for the most part. However, the front-end is inherently dependent on Terracotta, so modifying these features any further would require an understanding of Terracotta functions as well.

Through development of this project, we have arrived at a final product that meets the requirements and needs of our clients and serves as a strong first deployment of this tool. From the beginning of development to now, we have made modifications to how we implemented these features, mainly due to changes made to work with Terracotta. Initially, our design included Terracotta, but was more complex. After uncovering how powerful of a tool Terracotta is, we were able to work with it to develop simpler solutions to the hurdles we faced. One example of this is in exporting, as we initially were going to use a Google Drive implementation to provide downloads to source images. However, this would require maintenance of a copy of the source images on Google Drive apart from the actual source images being used for the application. Instead, we used Terracotta's method of keying and the file naming convention created for that process to trace back to the images, which is a much more centralized and straightforward solution.

While Terracotta made it possible to have a fully functional application ready by deployment time, using it for our application did have downfalls. One major aspect of our interface that we aimed to correct was the projection the map interface used. It currently uses Web Mercator projection, which is the standard for web-based maps. However, because our data is centered on Antarctica, and the Web Mercator projection distorts areas near the poles, it would be more intuitive to use a Polar Stereographic projection. This change was not able to be implemented prior to deployment, as Terracotta relies on packages based on Web Mercator, and thus, changing the projection would require extensive modifications to Terracotta. This, as well as other features we feel would improve the application, are discussed further in Future Work.

5 Testing

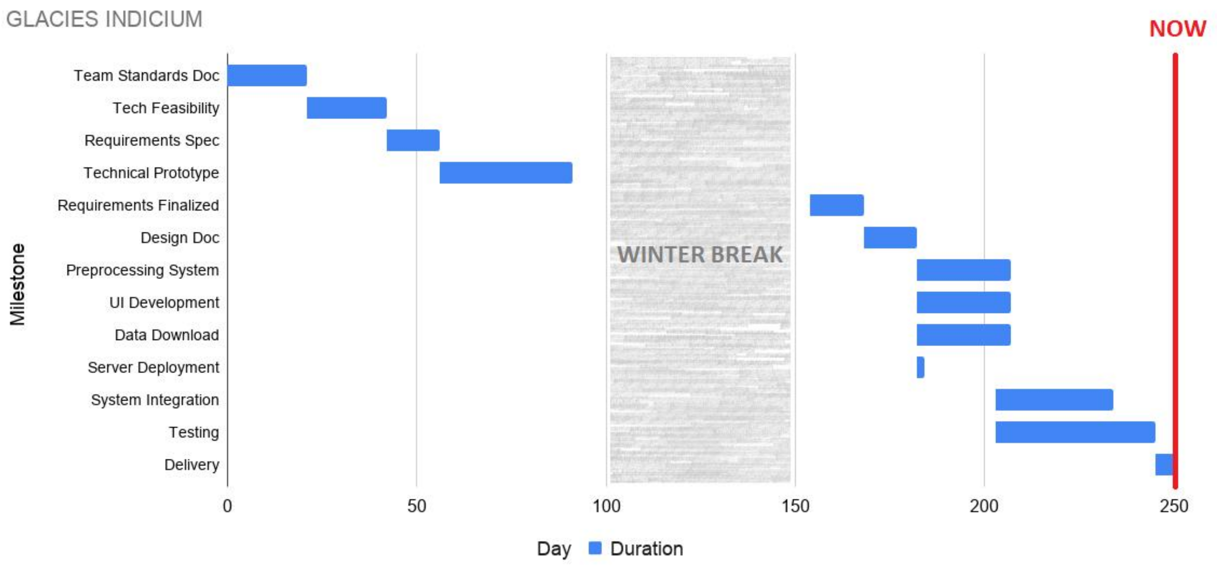
To ensure that our project meets our clients' base requirements and allow for expanding the database of source images our application visualizes, our specific software testing plan focuses on verifying the applications functionality, performance, and scalability. For each type of testing, we chose the most intuitive test plan for each component by working through our design from the bottom-up and implementing tests through software and libraries that are effortless to incorporate with the software components we have used in our implementation.

With the focus of our testing plan being on the preprocessing module, we aimed to bridge the connection between physical scientists and programming, by providing a preprocessing module that is easy to understand and execute. Through unit testing, we ensured functionality of base requirements for the preprocessing module, with the main goal of guaranteeing that this data will be able to be efficiently served to the front-end application with no errors or loss of data. On the processing side of the front-end, unit tests also helped to validate that this data is accessible by the visualization interface. Through this process, specifically, we were able to discover from the front-end that the preprocessing module was not able to generate all the regional mosaics for lower subregions and resolved this through minor changes in the mosaic generation.

The preprocessing and frontend module are connected seamlessly using the tile server Terracotta, which is why we created our software design with it as a main component. However, through our integration testing plan, we verified that the integrity of our data is preserved as it flows from preprocessing module to Terracotta to the front-end application.

Finally, usability testing was incorporated to confirm not only that we are meeting the requirements for a minimum viable product set forth by our clients, but more importantly, that our product provides a useful solution to their problem. Usability testing mainly confirmed our clients' overall satisfaction with the product, but it also revealed some extra components that could be incorporated in the future to further improve the design.

6 Project Timeline



The first few milestones of our project included working towards gaining a deeper understanding of our clients and their goals, researching how to best implement our solution, and specifying our findings in documentation for our clients to approve. This process aimed to reduce critical errors later in development and ensure that our product would meet our client's requirements. At the start, we created a Team Standards document to outline team expectations and ways to reduce and manage team conflict, should it be necessary in the future. For example, this document included expected response times, our attendance policy, and what each member's role was within the team. Conversations about these concepts through development of this document were crucial to the overall success of our project as it clarified expectations for each member of the team from the beginning.

Next, we worked to complete our Technology Feasibility document, which involved determining if our project was even possible to accomplish within the time frame we were given. To accomplish this, we broke down the requirements given to us by our clients and researched possible solutions for each of them considering our time constraints. Upon completion of this step in the development process, our team had created a general outline of our system and a rough schedule for design and implementation. We were able to move on the next step of implementation: understanding how different requirements might connect with each other.

Using a top-down approach, we worked from the general outline we had previously developed to detail more specific requirements. We broke our project down into domain, functional, performance, and environmental requirements using key information provided to us by our clients over the course of numerous regular meetings. Feedback from our clients on research conducted while working on our Technology Feasibility document allowed us to accomplish this and thus create our Requirements Specification document.

For our Technical Prototype, we were able to show others how far into our project we had gotten before holiday break. This presentation reviewed the work we had completed leading up to our complete software design and outlined our plan for implementation. This demonstration worked to gain feedback from our peers about our design and ensure that we would be ready to begin implementation when returning.

Implementation began with completely developing and finalizing the components related to our three major components: preprocessing system, user interface development, and data download. Each one of these was accomplished by different subsections of our team. Logan and Joe worked primarily on the preprocessing system, which takes in a folder of GeoTiffs given to us by our clients and returns a folder full of mosaics broken down by region and band. Zach worked on our user interface by making modifications to Terracotta's front-end to reflect our project's requirements. Kassandra and Beck were responsible for finding a solution to data exporting, which was accomplished by retrieving the stored GeoTiffs within Terracotta using the keys and filenames created in preprocessing and creating a corresponding download link for the source data.

After getting all the modules working independently, the next major step was to integrate them all together. This was accomplished by taking the mosaics created during preprocessing and testing them with Terracotta. After starting the server, functionality could then be examined through the front-end in the application, including functionality of the exporting feature.

After troubleshooting issues as they occurred, and working towards a product that seemed completely functional, we began testing to guarantee each component was working, even in the event of boundary cases and erroneous conditions. This was accomplished in three different steps: unit, integration, and usability. Unit testing was done on each method in each module, ensuring that everything worked correctly on its own. Integration testing was done on the application, this way when our clients when to use it there were not any problems for them to run into. Usability testing involved letting actual end users work with the application, this let us know what needed to change so user had a more enjoyable time working with our project.

7 Future Work

Our project serves as a prototype and minimum viable product upon delivery, and therefore has room to grow into the ideal tool for our clients. Through requirements acquisition and usability testing, we have identified a few specific features that would further improve our product.

Firstly, it would be most intuitive for the map visualization to use a Polar Stereographic projection, as the map would then be centered on our area of interest, the continent of Antarctica. Our current implementation is based upon the open-source tile server Terracotta, which uses the Web Mercator projection, a projection used in applications like Google Maps that is centered on the equator, and as a result, exaggerates areas near the poles. The use of Terracotta was essential to our product, as it provided the framework to ingest, serve, and visualize large amounts of raster images quickly, allowing us to focus on development of the preprocessing and frontend modules. We initially set out to change the projection for the initial deployment, however, through the development process, we became aware that Terracotta was extensively based on Web Mercator, and even utilized other packages and libraries based on this projection as well. Thus, changing the projection would require significant modification to Terracotta, which was not possible given our time constraints.

While not a requirement for a minimum viable product, we aimed to provide the option to view a RGB composition of the red, green, and blue single band layers. This feature would allow researchers to view a standard satellite image of how the Earth would look from above, in the range of color that we would see naturally. However, the bands served to the frontend needed to be split to single-band images as Terracotta converts the single-band raster images to a PNG format for use in the map application. PNG is the standard for web images and viewing the images in this format ensures quality performance of the application and reasonable load times with relatively large raster files. Because the images are in this format, however, forming an RGB composition on the front-end is not simple. One option for this would be to layer them on top of each other on the front-end by modifying the module in Terracotta responsible for creating Leaflet raster layers, but this may result in a visualization that is not a true RGB interpolation. Thus, the other option for this would be to create an RGB composition in the preprocessing step, but again, we were not able to implement a solution to this due to time constraints.

Lastly, our clients felt it would be helpful to be able to have the tool calculate statistics on the values within the raster images themselves. One feature that would result from this is the ability to query the images based on geological statistics. Essentially, this would mean for the user, that they would be able to search for geological components, and the images with a specified probability of containing those components would be displayed. It would also be beneficial to generate a statistical geological composition summary of a selected region. This would require significant amounts of further development and was not included in the requirements specification because of this. However, it would transform our application from a visualization tool to a scientific analysis tool, and therefore, greatly increase the power and possible applications for it.

8 Conclusion

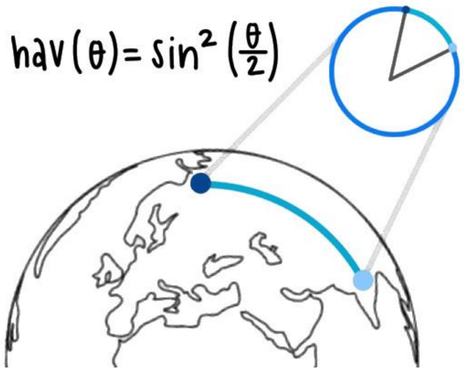
The hostile Antarctic environment presents a unique set of challenges to scientists trying to understand it. We aim to build an application that helps scientists and explorers in organizations around the world unravel the mysteries of this largely undiscovered frontier, expanding not only our understanding 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 exportable for use in offline contexts.

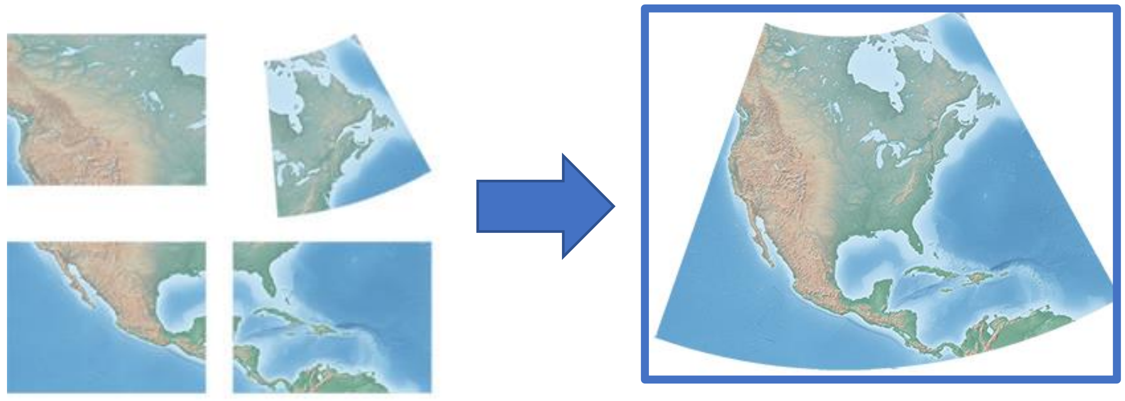
In summary, this project will allow geologists to view and filter geological data at the continental level in hopes of uncovering discoveries that will lead to new expeditions in the region. Overall, we were able to confirm functionality of base requirements necessary for acceptance and meet our goal of providing an easy, simple, and efficient way for our clients to navigate their data and further their study of the Antarctic.

9 Glossary

- **Haversine distance:** great-circle distance between two points on a sphere; used to find the shortest distance between two points on Earth

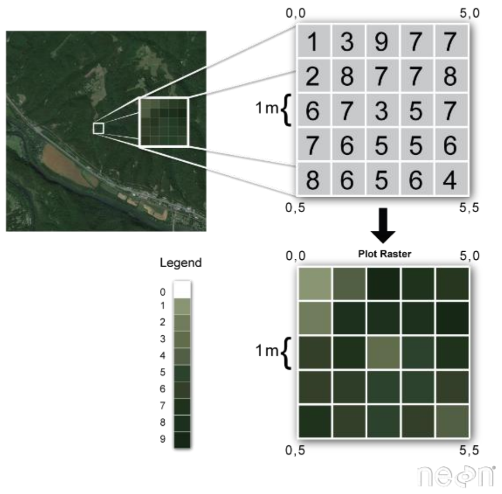

$\mathrm{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right)$

- **mosaic:** composition of multiple images to form one single, cohesive image



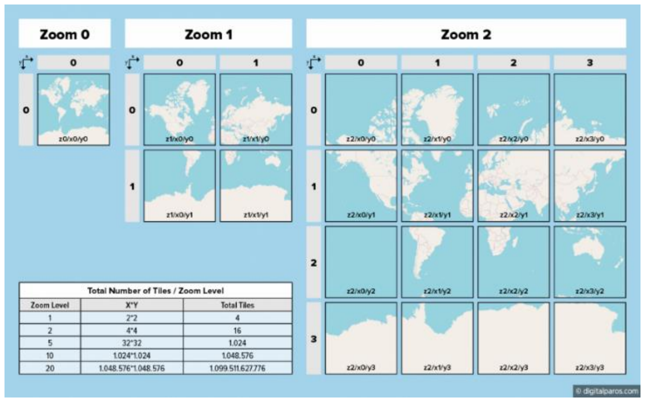
- **projection**: type of method used to flatten the surface of the Earth in order to represent its geography on a single plane

- **raster image**: representation of an image as a rectangular grid of pixels, with the color values at each pixel stored in a matrix structure.



1	3	9	7	7
2	8	7	7	8
6	7	3	5	7
7	6	5	5	6
8	6	5	6	4

- **RGB interpolation:** refers to the process of specifying the color space for an image as RGB (red-green-blue) which is the most common as it represents the color space visible to humans
- **XYZ tile**: a single cell in a grid created for a map interface which helps with placing images on the map at the proper coordinate location, navigating through the map, and rendering images at different zoom levels



Total Number of Tiles / Zoom Level

Zoom Level	X*Y	Total Tiles
1	2*2	4
2	4*4	16
5	32*32	1.024
10	1.024*1.024	1.048.576
20	1.048.576*1.048.576	1.099.511.627.776

Acronyms:

- **PNG**: Portable Network Graphic; standard image format for the web
- **GeoTIFF/TIFF**: file type used for raster images that contain geographical information

Appendix A: Development Environment and Toolchain

An incoming capstone team will want to familiarize themselves with the type of data their client is working with as well as confirm capabilities of viewing the data in a visualization software like ArcGIS. Understand that raster files are images made up of layers called bands and these bands can represent wavelengths, which can combine to make an RGB image, or modeled data like presence of snow or ice.

Most of our development was done on Linux but the product functions as planned on Windows. There aren't hard requirements in terms of hardware requirements besides being a functioning computer but know that once you are serving with a dataset of greater than a few dozen resource usage will begin to rise.

Toolchain

Preprocessing Script

- Primary languages used: Python
- Libraries used:
  - pathlib: for accessing local file directories
  - uuid: for generating random ids for each raster image without having to worry about conflicting names
  - json: used for writing and reading JSON data from local files.
  - emoji: used for printing UTF-8 emojis to the terminal when finished.
  - os: used for creating and removing temporary files during the preprocessing script.

User Interface

- Primary languages used: HTML, CSS, JavaScript, jQuery as they are fundamental for building any web application and jQuery provides cleaner functions that vanilla JavaScript.
- Libraries used:
  - Halfmoon CSS: for providing HTML structures such as a side bar and navbar using the same format as Bootstrap but with extra functionality such as dark mode.
  - Leaflet: for creating the interactive map that the raster images are displayed on.
  - NodeJs: used for creating unit tests for the front-end module.

Backend-Frontend Integration

- Library used:
    - Terracotta: Terracotta provides a simple way to ingest a large set of raster images and quickly deploy them onto an interactive map using Leaflet.

Setup

This setup process has been tested on both Linux/Unix systems as well as Windows to provide easier access to the application.

- First, install Python which can be found at: https://www.python.org/downloads/.
- Next, install Anaconda to run Terracotta which can be found at: https://www.anaconda.com/products/individual.
    - After this is installed verify that it installed properly by first opening the program.
        - Windows: Click Start, search, or select Anaconda Prompt from the menu.
        - macOs: Cmd+Space to open Spotlight Search and type "Navigator" to open the program.
        - Linux-Ubuntu: Open the Dash by clicking the upper left Ubuntu icon, then type "terminal"
    - After Anaconda is open, run the command `conda list.` If Anaconda is installed and working, this will display a list of installed packages and their versions.
- After Anaconda is installed, run the below commands to install GDAL and rasterio which are used for reprojecting the raster images.

```
>> conda install -c conda-forge gdal
>> gdalinfo –version
```

  If GDAL was installed properly, the above command should output the current version of GDAL.

```
>> conda install -c conda-forge gdal
```

- The next step is to install Git to clone the current repository for this project which contains the preprocessing script and the user interface. Git can be installed using the following link: https://git-scm.com/downloads
    - Then verify Git is installed by running the command

        - `>> git –version`
    - If the current version of git is reported back to you, clone the repository from this link https://github.com/mlgarrett/glacies-indicium by using the command:
        - `>> git clone https://github.com/mlgarrett/glacies-indicium`
- After the repository finishes downloading, change into the repository directory and create the Terracotta environment using the following commands:
    - `>> cd glacies-indicium`
    - `>> conda env create -f environment.yml`
- After the environment is created, activate the Terracotta environment by running the following command:
    - `>> conda activate terracotta`
- The next step is to run the preprocessing module on a set of raster images which will generate the necessary single band images for the Terracotta client. The preprocessing script requires that a path to raster images is provided when ran otherwise it will not work properly. An example of this command can be seen below:
    - `>> python raster-prep.py -d <input folder path> [OPTIONS]`

  The optional command `-r` or `--remove_image_border` can be appended to the command above which will remove the borders around the raster images. As we discussed in client meetings, the no data values will mask real data in the raster images unless this is accounted for in the clients processing pipeline. Otherwise use the remove image flag as seen below:
    - `>> python raster-prep.py -d <input folder path> -r`

- After the preprocessing module is finished, the next step is to optimize the raster images and ingest them using Terracotta. More information about this command and other Terracotta commands can be found at this link, https://terracotta-python.readthedocs.io/en/latest/cli-commands/serve.html. Using Anaconda in the Terracotta environment in the root directory run the following command:
    - `>> terracotta optimize-rasters terracotta\client\static\mosaics\*.tif -o terracotta\client\static\mosaics\optimized\ [OPTIONS]`
    - After the above command finishes run the following command to ingest and serve the raster images:
        - `>> terracotta serve -r terracotta\client\static\mosaics\optimized\ {region}_{band}.tif`

- The final step to starting Terracotta locally is to open a second Anaconda prompt. In this new prompt, go to the glacies indicium repository directory and run `conda activate terracotta`. Then run the next command to connect to Terracotta:
    - `>> terracotta connect http://localhost:5000`
    - This command will automatically launch the user interface in a new browser tab with the available optimized raster images.
- Additionally, Terracotta provides three API endpoints for further exploration of the available data. These URL links can be found below:
    - http://localhost:5000/keys
    - http://localhost:5000/datasets
    - http://localhost:5000/apidoc

Production Cycle

- The preprocessing module does not require any compiling before running, and simply requires the `raster_prep.py` file to be called in order to run.
- To edit the HTML in the user interface and view the changes, Terracotta requires a full restart in order to render the new changes. This can be achieved by running CMD/CTRL + C in the Anaconda prompt and then re running the same command from above.
- The JavaScript also does not need to be recompiled through Terracotta before running, but does require the user interface to be refreshed in the browser by pressing F5 to run the new changes.
- To deploy the changes, separate branches are used to push updates which can then be reviewed before being merged into the master branch.