Glacies Indicium

Software Design Document

Version 2

2/14/2021

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	

1 Introduction 3

2 Implementation Overview 5

3 Architectural Overview 6

3.1 Introduction 6

3.2 Database Backend 7

3.3 Terracotta Server 7

3.4 Terracotta Client & Custom Leaflet Map 8

4 Module and Interface Descriptions 8

4.1 Preprocessing 8

4.2 Backend Module 13

4.3 User Interface 14

5 Implementation Plan 18

6 Conclusion 19

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 its harsh climate, Antarctica remains extremely inaccessible to researchers. 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 the environmental conditions prevent researchers from conducting many studies there. However, as satellite imagery technology has developed, it has greatly facilitated our ability to study this tough 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. These images have an initial accuracy where each pixel is ~10 m x 10 m in the real world and each cover about a hectare of land. 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.

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, also called a GeoTiff or tiff after the file extension used for them: .tif. To illustrate what a raster image is, consider a normal digital image on your computer, which is made up of three bands: a red band, a green band, and a blue band. These three bands are added together to produce the full range of colors, resulting in a typical RGB digital image. A raster image, though, is not limited to a specific number of bands, and can also map these bands to places on the earth using geographical coordinates.

Some of the raster 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 where they run the raw spectral data through processing which filters out atmospheric anomalies to compute and derive bands of data representing geological 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 building our geological understanding of Antarctica, a continent that experiences some of the most extreme climate conditions on Earth, allows us to formulate scientific projections about other geological and environmental processes in other areas of the planet. In some cases, for example, these studies can showcase how Antarctica is not being as affected by climate change as other regions in the world. Currently, this research process is hindered by technological barriers that we aim to solve. Our clients are examining these multilayered satellite images layer-by-layer, one image at a time, which can individually visualize about 5 km^2. The first problem is that it is easy for a scientist to examine one image, which in size can be about 2 GBs, but it is difficult 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 do not have a means to filter 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.

2 Implementation Overview

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 filter data by actual geology, which the current technologies our clients work with do not satisfy on the continental level. 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 retrieve and display images raster data by the second and display a progress bar for load times longer than a few seconds. Python will be used to implement this project along with specific libraries like Terracotta, which provide 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.

3 Architectural Overview

3.1 Introduction

The architecture for this project will be divided into three different parts consisting of the database backend, the Terracotta server that acts as the connection between the backend and the front end, as well as the Terracotta Client and the Leaflet map graphical user interface. The figure below depicts a model of this complete architecture and how the components will connect.

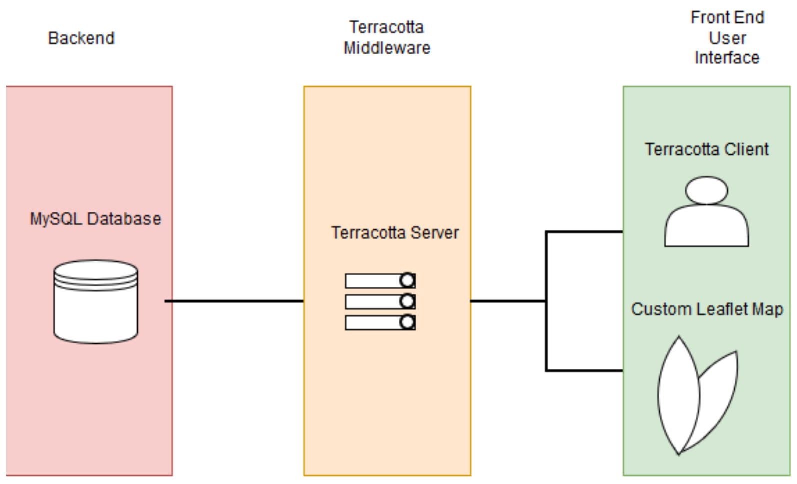

Figure 1 - Complete architectural overview of the website.

3.2 Database Backend

The backend for this project will consist of a MySQL database that will be used to store the several hundred raster images that will be presented to the user on the front end. Figure 2 below provides an overview of how the database will be constructed using different tables for higher and lower resolution images to increase the overall performance in the application.

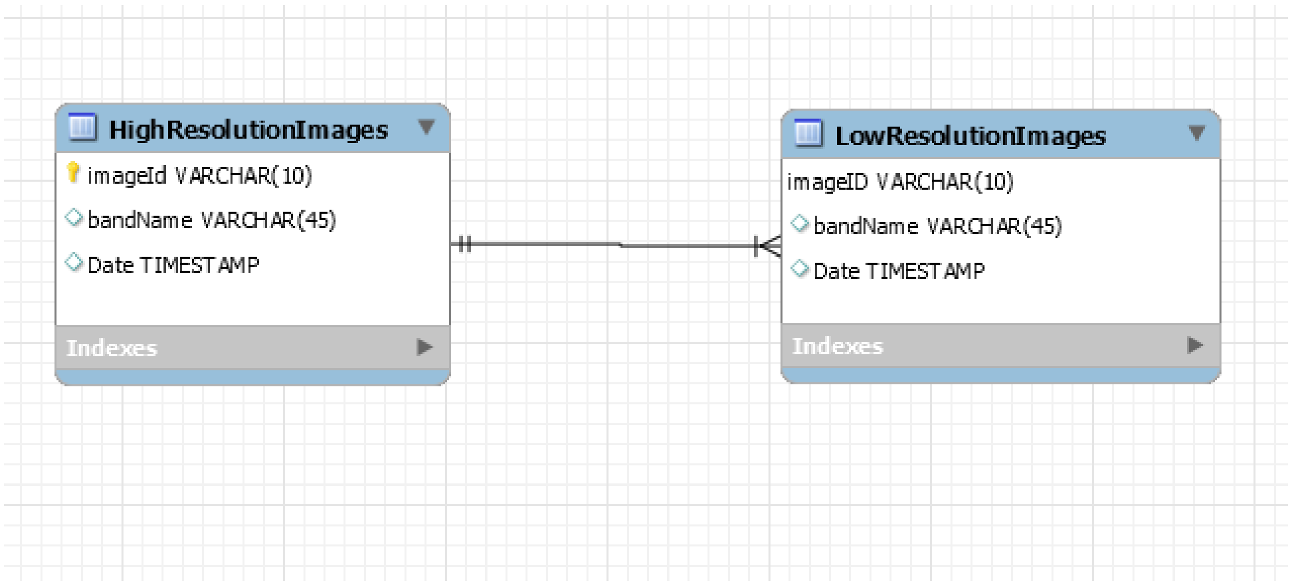

Figure 2 - Current database model separating the images into different resolution tables.

3.3 Terracotta Server

The Terracotta Server is the first script to be run before the user's webpage is loaded. The server will preprocess the data and create the necessary bands before the map is fully loaded. After the files have been preprocessed and separated into individual bands and potentially downsampled, the resulting PNG images will be sent to the Terracotta client to display onto the map. The biggest benefit of using Terracotta is that it acts as a base which is already able to display raster images to a map. This allows us to build off of Terracotta to provide the functionality our client is looking for in terms of filtering large amounts of raster data.

3.4 Terracotta Client & Custom Leaflet Map

After the data has been preprocessed and organized in the database, it will be sent to the Terracotta client to display as an interactive map. The Terracotta client will receive the individual band names, along with the keys used in the file name structure and present it onto the user interface. The user will then be able to click on each band name to display certain regions within the map. The map will also be heavily customized for this project in order to deliver a better projection of Antarctica. Currently, Leaflet uses a Web-Mercator projection of Earth, which makes Antarctica a rectangular shape and hard to view on a map. In order to present a better user experience, this projection will be changed to Polar Stereographic which will present a map that is better suited for this project due to being centered on Antarctica.

4 Module and Interface Descriptions

4.1 Preprocessing

The preprocessing module's primary goal is the organization of the input data our client gives us. Currently, we will receive 100-200 multiband tiffs that are composed of the reflectance data and geological data that we have described previously. However, the base that we are using to display tiffs to the user, Terracotta, can only ingest single band images. Additionally, a single tiff image can cover a relatively small area. We also have a regional nested hierarchy that the tiffs need to be organized into. Therefore, the preprocessing module is responsible for splitting each tiff up by band, keying the information by region and band via the file name, and combining smaller, adjacent tiffs into mosaics, which are merges of two or more images to form one larger continuous one. This last point is particularly important as one of our key problems we are trying to solve for our client is speeding up the pace at which they can gleam patterns from such a large dataset.

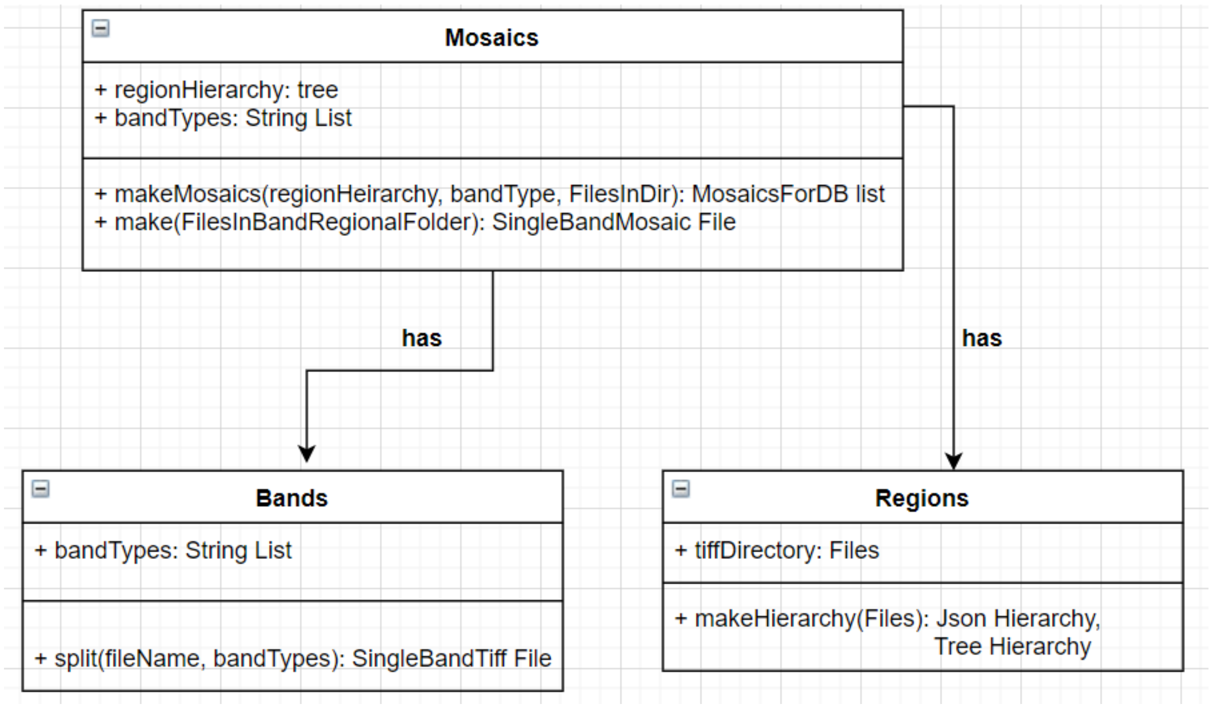
Mosaics
+ regionHierarchy: tree
+ bandTypes: String List
+ makeMosaics(regionHeirarchy, bandType, FilesInDir): MosaicsForDB list
+ make(FilesInBandRegionalFolder): SingleBandMosaic File
has
has
Bands
+ bandTypes: String List
+ split(fileName, bandTypes): SingleBandTiff File
Regions
+ tiffDirectory: Files
+ makeHierarchy(Files): Json Hierarchy, Tree Hierarchy

Figure 3 - Preprocessing class diagram illustrating how the mosaics will be associated with their respective bands and regions.

As you can see from Figure 3 the end goal of this module is to have a series of mosaics that represent distinct regions in Antarctica. Due to the constraints of the base we are using, each regional mosaic needs to be a single band of raster data.

Regions Class

The regions class will look through the clients input directory of tiff files and create a tree structure from the naming convention we have agreed upon. For reference, each single image fits in up to four subregions. The first encompasses all the Northern Victoria Land which can be thought of as the parent node A in Figure 4 below. This first region will eventually have multiple child subregions (for the purposes up to the Alpha Prototype we will only be working with one: The Dry Valleys). The Dry Valleys can then be divided up again and those subregions have subregions. For this purpose, we have agreed upon the following file naming convention for input tiffs from our client: *region1_region2_region3_region4_date_catID.tif.*

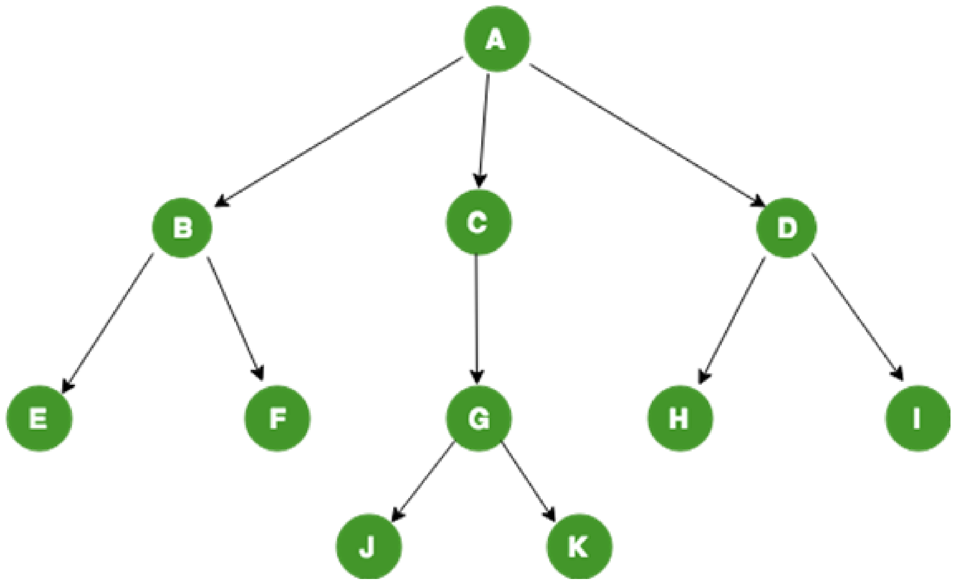

Figure 4 - Regions hierarchy diagram.

Each subsequent region will be a more deeply nested node on the tree. This may seem complicated but by accomplishing this now it will allow our clients to further expand the input data after the course of this project. For further preprocessing, this tree structure will be created but for use on the front end a version that will be written in JSON will be generated to aid in the display of this list of nested regions on the frontend.

Regions Class Fields

This class will only have one field that is necessary and that is the input directory of the source data our clients are giving to us. Using the naming convention described above we will be able to generate nodes in the tree based on what name a tiff falls. For example, the reading of

NorthernVictoriaLand_DryValleys_AsgardRange_MtNewall.tif & *NorthernVictoriaLand_DryValleys_AsgardRange_RoundMountain.tif*

would result in the creation of the tree structure shown in Figure 5.

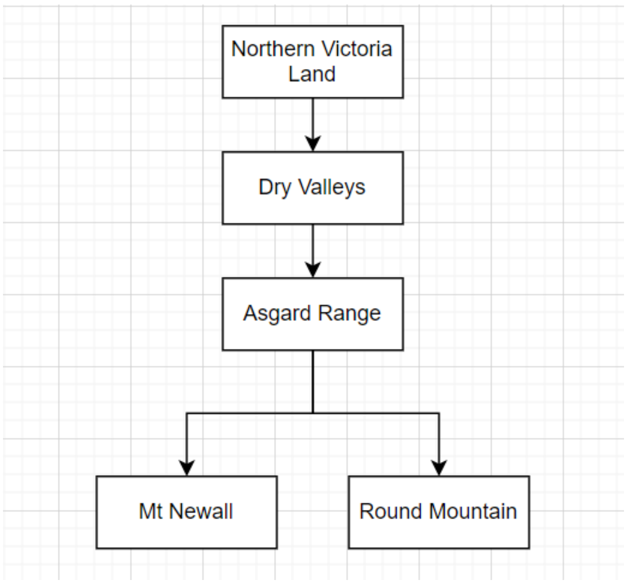

Figure 5 - Naming convention hierarchy diagram.

Regions Class Methods

The *Regions* class only has one public method and that is to generate the hierarchy that has been already described.

Bands Class

The bands class' main goal is to split up the multi-band tiff file. The class follows the procedure of *Regions* generation and takes in the same input directory of source files that the previous class did. The class works its way through each tiff outputting the single-band tiff in its' corresponding band folder

i.e., The *reflectance1* band for each tif gets put in the *reflectance1* directory, the snow band in the snow directory, etc.

Bands Class Fields

The *Bands* class has one primary field, and it is a list of the band names. This will be easily identifiable by our client should they need to change the number of bands the data will have as it is relatively variable. We are expecting to be working with less than a dozen bands.

Bands Class Methods

The *Bands* class has one primary method which is the split method. It will split off the specific band of the tiff file into the respective folder of the same name.

Mosaics Class

The *Mosaics* class will use the output of the previous class' work to create single-band mosaics of each specific subregion. The current plan is to go through each newly filled band folder and for each node in the tree create mosaics with the corresponding region name in the correct place in the file name.

i.e., All *reflectance1* tiffs with *NorthernVictoriaLand* in the *region1* placeholder will be merged into a *NorthernVictoriaLand_reflectance1* mosaic and repeating for every subregion, and every band. These mosaics can then be requested by the user through the frontend and exporting components of the application.

Mosaics Class Fields

The *Mosaics* class will take in the newly generated single band tiffs as well as the tree hierarchy.

Mosaics Class Methods

The most basic of methods will be the make method which takes in a subregion name, and a subregion level corresponding to the depth in the tree and will be called in a single-band directory. The make method will match tiffs that have corresponding subregion names at the correct part of the file name corresponding to a specific level in the tree.

. i.e., All single-band tiffs that have *NorthernVictoriaLand* before the first underscore will be made into a mosaic placed into the *0depth* folder. Each subsequent depth folder will have more mosaics as there are nodes in the tree.

4.2 Backend Module

The backend module for our project is used mostly at the start of deployment. Its main job is to ensure that Terracotta starts up correctly on a web page and allows the user to choose either to start over with a new data set or restate from a pre-existing server. If the user chooses to start with a new database, they will need to be able to identify their new database and run the preprocessing module on that database. Whereas, if an existing set of data is used, then the prepossessing would already be done.

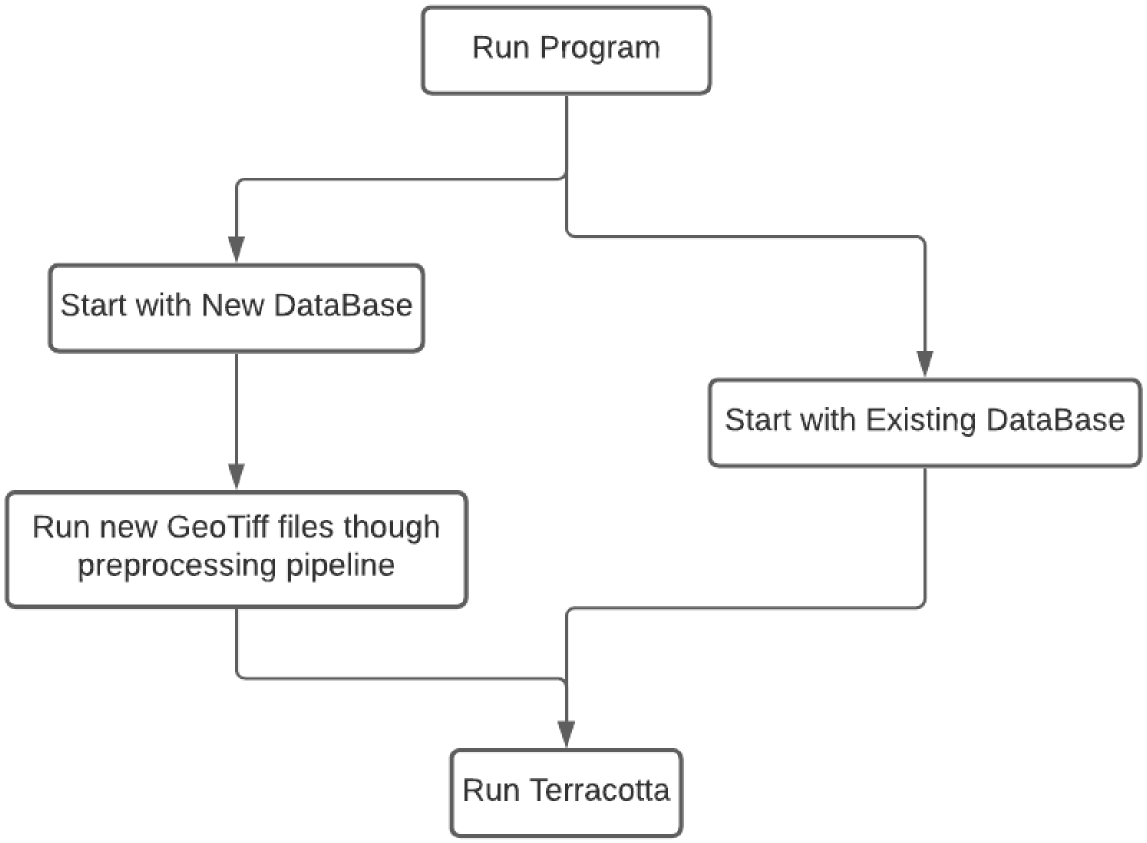
Run Program
Start with New DataBase
Start with Existing DataBase
Run new GeoTiff files though preprocessing pipeline
Run Terracotta

Figure 6 - Server startup diagram that illustrates how the server can be started in two different ways – through a new database or an existing database.

This diagram in Figure 6 shows that there are two different ways the server can be started up. On the left side, there is a new database that has never been used before and it takes more steps to get it online. Whereas on the right with a pre-existing, all it takes is selection of the previous database.

4.3 User Interface

The user interface module will be primarily responsible for modifying the Terracotta user interface and adding new functionality such as a nested view of all the available regions. This module will handle any interaction from the user and will modify the interface as needed or send information to the database backend to retrieve new information. The UML class diagram can be seen below which will create a progress bar, create the region dropdown list and listen for event changes such as a click or scroll within the map.

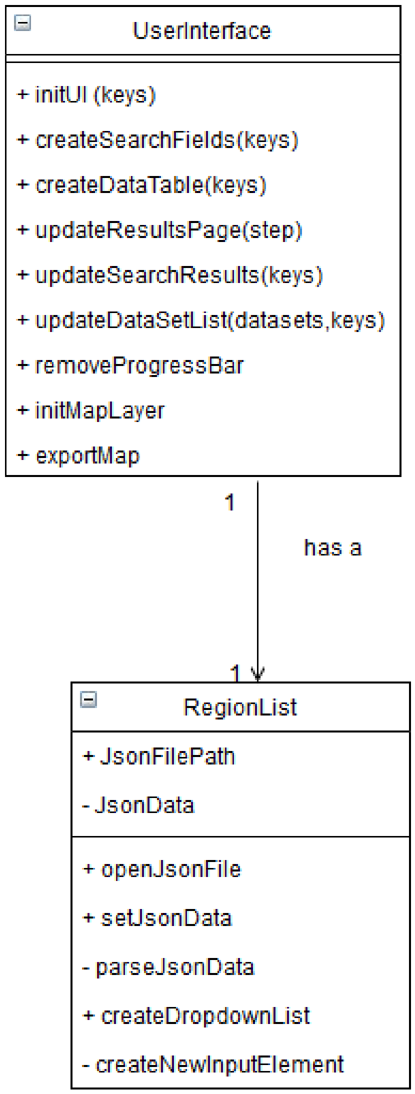

Figure 7 - UserInterface class diagram detailing the methods datatypes this module will require.

UserInterface Class

The public interface for the front end of the application is the *UserInterface* class which also serves as an entry point to the *RegionList* class. The *UserInterface* class is also responsible for handling any event changes to the input forms within the application as well.

UserInterface Class Fields

The *UserInterface* Class does not contain any class fields as it is only responsible for listening to event changes within the page and retrieving needed information from the Terracotta Server and Client.

UserInterface Class Methods

In order to provide functionality to the user several class methods will be used to listen for changes within the page. These changes will be detected and handled by the *updateResultsPage*, *updateSearchResults,* and *updateDataSetList* methods.

When the user changes a region from the region dropdown menu, the *updateSearchResults* method will be called and will render a new map of the selected region back to the user. This will also update the dropdown menu through the *updateDataSetList* method to make the selected region text bold to notify the user of what is currently being displayed on the map.

The *updateDataSetList* method is used to change what regions are displayed on the user interface. This method will also determine the current page the user has selected via the pagination form below the dropdown menu.

The last method, *exportMap* is responsible for listening to a click on the “export map” button on the user interface and take a screenshot of the current map and save it locally to the user's machine. Leaflet has several plugins available to capture screenshots, which will be used to provide this functionality to the user.

RegionList Class

The *RegionList* class is primarily responsible for getting the available regions from the dataset and parsing the information in order to display a dropdown menu of the regions within the user interface.

RegionList Class Fields

Currently, only two fields are necessary in this class which are the public *jsonFilePath* and the private *jsonData* fields. The *jsonFilePath* field is a string to let the class know where to locate and open the JSON file. The *jsonData* field will be stored as a JavaScript Object Notation (JSON) object with data from the JSON file which allows for the front end to quickly parse the list of regions to display.

RegionList Class Methods

In order to determine the available regions, the JSON file must be opened first before parsing which will happen within the *openJsonFile* method. This method will open the JSON file specified within the class and will use the *setJsonData* method to initialize the *jsonData* field to be the data from the open JSON file.

After the JSON file has been opened and the *jsonData* field is initialized, the *parseJsonData* method will be called to iterate over each region. Every region found within the *parseJsonData* method will pass the data to the *createDropdownList* method which will be used to create the main menu and an input element for each region using the *createNewInputElementMethod*.

5 Implementation Plan

The outline of our team's implementation plan is detailed below in Figure 8, which depicts our plan for the remaining tasks in the semester. Currently, our team is divided into two separate teams who are working on early development for the Terracotta backend and the technical writing for written assignments. After the Software Design document is finished, the team will be focused on finishing the early prototype for the second design review along with testing the current implementation to ensure the programs stability and effectiveness at delivering the best possible solution. The final projects for the team are to implement needed changes from the software testing and the second design review to present the final project in both the third design review as well as in the capstone project poster at the end of the semester before the completed project is delivered to the clients.

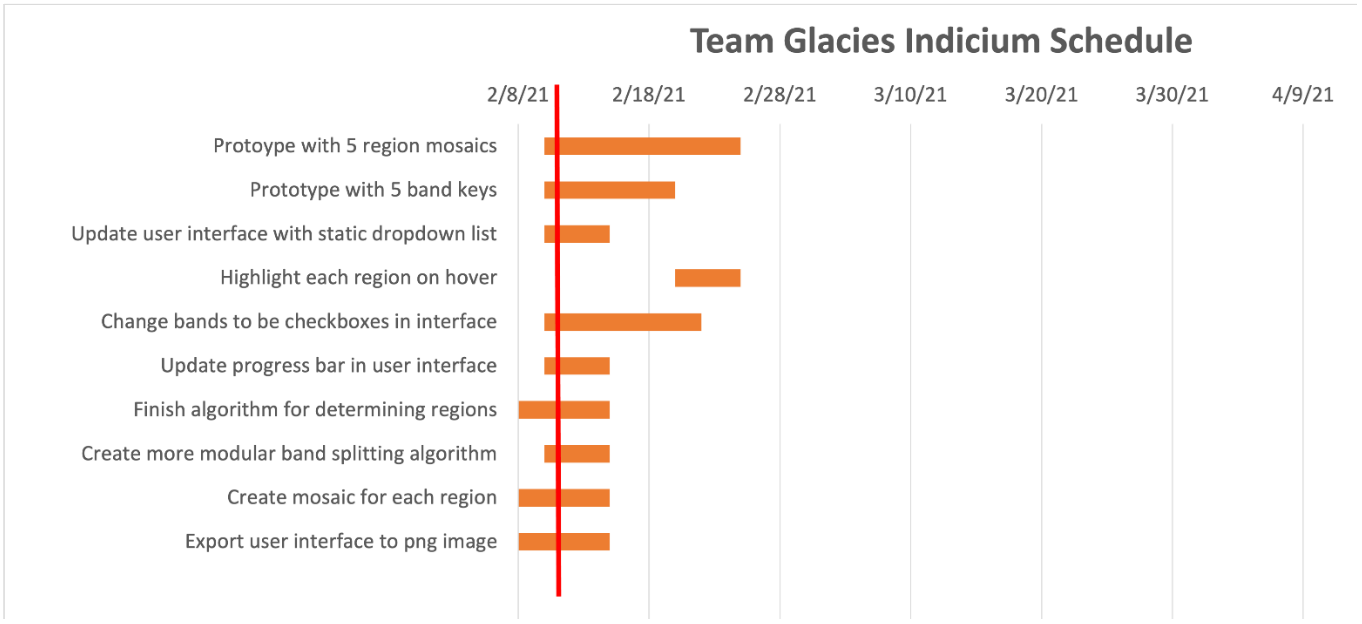

Figure 8 Current team schedule up until delivery of Alpha Prototype, red line represents current day.

6 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.

The purpose of this document has been to establish how our team plans to implement this interactive map of Antarctica. 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 content at the conclusion of the project.

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. To give a short update of current process, our team is currently receiving access to the pipeline of images our client has for us to work with and work on both the preprocessing pipeline and user interface is well into production.