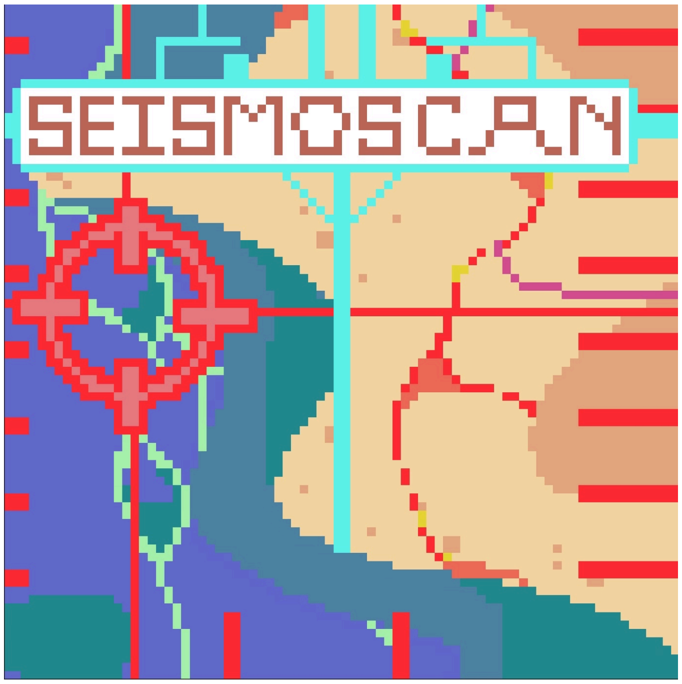
Requirements Document

SeismoScan

Sponsors: Dr. Donna Shillington, Dr. James Gaherty, Dr. Christine Regalla



Mentor: Scott LaRocca

Alyssa Sombrero

Brady Wisniewski

Noah Carges

Thomas Rotchford

11.26.2025

Team Lead: [illegible] Version: 1.2.0 Client: [illegible]

2. TABLE OF CONTENTS

2. TABLE OF CONTENTS........2
3. INTRODUCTION........3
4. PROBLEM STATEMENT........4
5. SOLUTION VISION........5
6. PROJECT REQUIREMENTS........6
6.1 Functional Requirements:........6
6.2 Performance Requirements:........9
6.3 Environmental Requirements:........10
7. POTENTIAL RISKS........10
8. PROJECT PLAN........12
9. CONCLUSION........12

3. INTRODUCTION

In the field of geology and seismology, there is a need to understand Earth's structure and natural hazards. The geoscience industry is valued in thousands of dollars, which are invested to support environmental assessments, earthquake preparedness, tsunami mitigation, exploration for renewable and non-renewable resources, water management, and more. Within the vast, data-driven industry, fault identification and tectonic analysis are an essential and fundamental process for academic research and applied to the industry's operation.

Our sponsors, Professors Donna J. Shillington, James B. Gaherty, and Christine Regalla from Northern Arizona University's School of Earth and Sustainability are active researchers in deformation, magmatism, sedimentation, and earthquake seismology. They have published works that contribute to the understanding of crustal formation, plate tectonics, and seismic activity, which relates to the objective of the project they have given us - creating a machine learning model to map faults in bathymetry data. This model will save the sponsor's time and enable more consistent fault mapping, and thus contribute further research on earthquake studies and tectonic plates.

The main problem our sponsors must solve is the time consumption it takes for a research assistant to manually identify faults by the given bathymetry data. This time-consuming process involves visual interpretations and characterization of fault attributes, and thus requires valuable research time and limits productivity. Furthermore, another downside of manual fault interpretation is that it could introduce potential inconsistency in the fault maps in terms of the level of detail of mapping.

We will develop a machine learning model to improve the efficiency of our client's workflow by producing more reliable and accurate data for use in their published research. By significantly reducing processing time, from weeks to minutes. The model will accelerate the client's ability to analyze seismic data in a deeper state, compared to what manual mapping can offer. Thus the machine learning model will make the workflow faster, consistent, and efficient.

By integrating machine learning to the seismic workflows, the project aims to improve the accuracy, speed, and consistency in fault identification. Ultimately, it will enhance the productivity of our sponsor's research team while advancing in the geoscience industry's pursuit of data-driven insights into Earth's fault system.

4. PROBLEM STATEMENT

The current workflow used by the sponsor's research teams involves manual fault mapping. The researcher, associate, or student starts by visually inspecting the given bathymetry data typically as NetCDF or other geospatial data format. Then they perform the mathematical analysis through slope and depth calculations and comparing the potential fault with known fault attributes. From the information they gathered on other potential locations, they determine if the location is a fault, or a false positive fault. They continue repeating this process until they are confident that they mapped all the potential faults. They report their findings by marking the spots where the faults are found and export their findings for other researchers to peer review.

However this method is time-intensive, subject to inconsistencies and difficult to standardize across multiple datasets and researchers as listed in the following:

Deficiencies and Missing Elements

- **Excessive Time Consumption:** Mapping faults manually can take hours or days per given bathymetry dataset, slowing down the overall research progress.

- **Inconsistent Results:** By human judgement, it introduces variations and errors in fault interpretations, therefore reducing the reproducibility and reliability of data. Researchers may overlook small, subtle faults or produce maps with inconsistent levels of detail of fault mapping (picking only large faults in so…

The inefficiencies listed highlight the need for a machine learning-based automation system that can process the input data, identify potential faults, and output the coordinates with minimal human intervention. By incorporating the system, the sponsor's research team is able to have an efficient workflow, data accuracy, and analytical depth.

5. SOLUTION VISION

In order to provide a well-rounded solution we will build a command line interface that intakes bathymetry data and outputs latitude and longitude points that are located on or within a fault, using machine learning models as our method of detection. Here are the features that are required in order to deliver this software:

- A trained machine learning model that can detect faults from bathymetry data
- Set of tools to isolate important parts of the bathymetry data for easier understanding and use within the model
- Ability to output to a text file that provides latitude and longitude data of the faults
- Comprehensive documentation to allow for easy usage

In order to complete this project we will need multiple sets of data. A data set that has been labeled will be split to be used as training data, validation data, and testing data. Along with that other unannotated bathymetry data will be provided in order to simulate usage scenarios. This data will be formatted as NetCDF data and provided to the team by the clients.

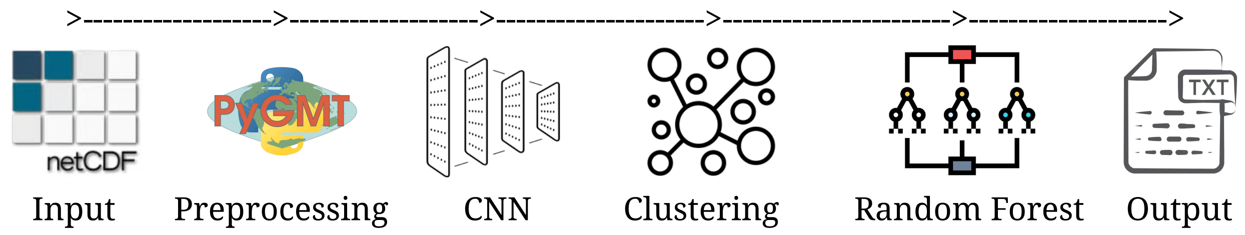
To output the models conclusions, we will be storing them as a simple .txt file containing latitude and longitude data points. The first value is latitude and the second is longitude. Each point will be separated by a new line.

There will be two main computational processes done on the backend of this project. The first will be a set of data cleaning and formatting tools in order to remove unnecessary data and calculate useful values such as slope with the data before providing it to the machine learning model. Second will be the machine learning model itself which is currently planned to hybrid pipeline approach including a Convolutional Neural Network, Clustering algorithm, and Random Forest*.

This machine learning approach will allow geologists like our clients to be more time efficient and interact more with the processes that these faults affect rather than time spent identifying them by hand. This will allow for a better understanding of the earth's deep water processes which are believed to affect things including earthquakes and volcanic activity.

*Subject to change

Dataflow



6. PROJECT REQUIREMENTS

In order to successfully complete this project, we need to deliver a simple interface that connects to a machine learning model that will map faults leading to an improvement in geological research.

6.1 Functional Requirements:

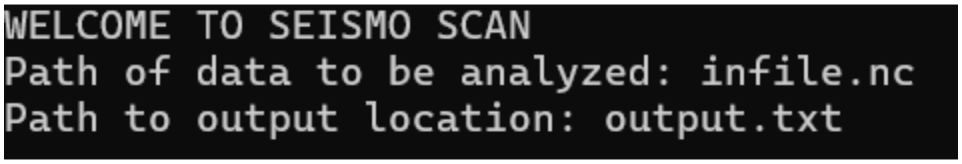
Below we present a table of the functionality based requirements for this project. They are grouped into 4 categories including:

- Must: All must be present to reach a MVP stage
- Should: While not absolutely necessary, these will be implemented to add real value to the project, ordered by importance
- Could: Stretch goals which will be completed should time permit
- Wont: Specifically not necessary and do not want in this project

Must	Should	Could	Wont
- Command Line Interface - Ingest NetCDF data - Machine Learning Model - Output the Latitude and Longitude to a .txt	- Config File - Provide confidence level in .txt file - Provide Debug/Log files - Installation Wizard	- Model sensitivity control options - Fault Attributes - Graph integration	- UI/UX - Database - Immediate response

6.1.1 Command Line Interface - Must:

The first must is to build a command line interface that provides a location of the files needed for the program to correctly run. This should be able to take in an absolute path as well, to allow it to take in or output data into a file located separate from the program.



```
WELCOME TO SEISMO SCAN
Path of data to be analyzed: infile.nc
Path to output location: output.txt
```

6.1.2 Data Type - Must:

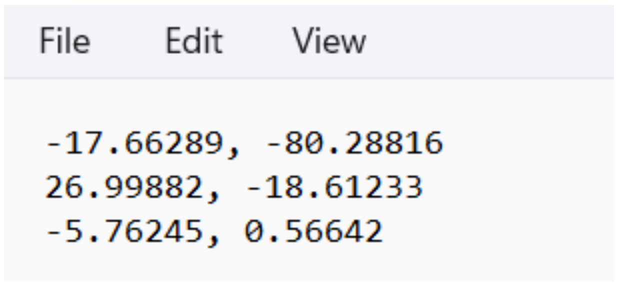
As for input data it is provided exclusively as NetCDF data formatted in a .grd file. This data will need to be preprocessed and stored as seen fit to then provide to our machine learning model.

6.1.3 Machine Learning Model - Must:

A machine learning model will be developed that will connect on the backend to take in preprocessed data. This model in its simplest form will use a variety of algorithms to detect and label different latitude and longitude points as part of a fault.

6.1.4 Output - Must:

To finish in its simplest state, the model will then relay points it believes to be faults. This data will be stored and printed in a .txt file in the following format. Latitude, Longitude. Each point will be displayed on its own line with a comma separating the latitude and longitude values. Each fault will be separated by a delimiter of “>” to provide clarity as to where a new fault begins.



```
-17.66289, -80.28816
26.99882, -18.61233
-5.76245, 0.56642
```

*This does not represent real fault locations and is randomly generated numbers

6.1.5 Confidence Level - Should:

The most important but non vital attribute is the ability to provide a confidence level of each identified point. This will be information taken specifically from the model and provided as a percentage in a separate but similar file that would include all other information. This will allow the user to more efficiently hand check the work of the model by focusing on the less confident results while being able to mostly ignore the high confidence faults.

6.1.6 Log Files - Should:

Similarly providing an option to generate an extensive debug/log file that provides more in depth decision making information from the machine learning model. This will allow for continued refining of the model and better understanding of its decision making at the completion of the project in order to better identify its quirks. There will also be an option to have verbose in the command line that will contain the log file information.

6.1.7 Installation Tools - Should:

Finally an installation wizard would be provided. This would provide a simple and comprehensive way to install all dependencies in the necessary fashion to enable the project to correctly run. This allows for ease of access should this tool be used in the field on portable machines.

6.1.8 Sensitivity - Could:

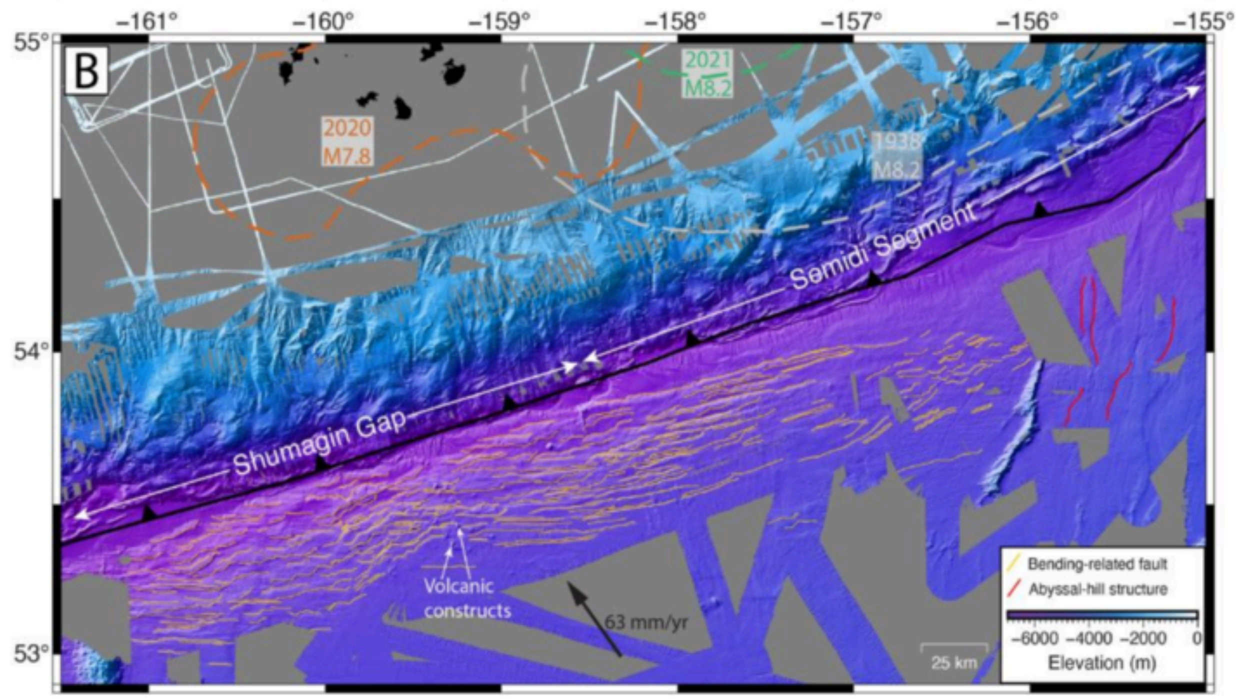
The first stretch goal for this project will be to add options to the configuration file that allow tuning the sensitivity of the different models to allow for a higher level of fault detection for the trade off of more false positive values. On the other hand it could be changed to give less false positives for missing more faults.

6.1.9 Fault Features - Could:

The next would be adding some other fault features and information to the file containing the aforementioned confidence data. This would include information such as the Slope, Azimuth, elevation offset, dip direction of the faults it detects.

6.1.10 Visual Integration - Could:

The final stretch goal is to include a level of integration with GMT that would allow to display these faults on a visualized map, saved as a png file. An image similar to what this might look like is included below.



*This image includes data that would not be part of the final product

6.1.11 Will not include:

There will not be any time spent on building a user interface and implementing any user experience features. There is also no database required to complete this project as all data will be uploaded to the model at runtime from the local machine. This will not be a project that is required to have a quick response time. Due to the large data sets, the project may take a large amount of time to calculate the necessary information and should not be expected to have immediate response.

6.2 Performance Requirements:

In regards to performance, we will be describing the requirements that allow our program to effectively perform its tasks, focusing on speed, accuracy, and overall usability compared to the current manual identification process used by geologists.

6.2.1 Processing Efficiency:

Minimum Benchmark Requirement:

- The system **reduces fault-detection time by at least 80%** compared to manual mapping performed by geologists.
    - Ex: If I dataset typically requires 5 hours to review manually, our model should process it in 1 hour or less

Target (Desired) Benchmark:

- The system should process typical research-scale datasets in **under 15 minutes** when running on standard lab hardware

6.2.2 Accuracy:

Minimum Benchmark Requirement:

- The model correctly identifies at least **50%** of true faults present in a dataset.
- No more than **30%** of all detected faults shall be false positives.
- The system **outputs a confidence level** for each detected fault to support user validation.

Target (Desired) Benchmark:

- Identifies **70%-80%** of true faults present in a dataset.
- No more than **20%** of detected faults may be false positives

6.2.3 Reliability:

Minimum Benchmark Requirement:

- The system completes **100%** of analysis runs **without crashes** when provided valid, correctly formatted datasets.

- Given the same dataset and configuration, repeated runs produce outputs with no more than **5% variation** in detected fault locations (due to randomness in ML models).

Target (Desired) Benchmark:

- The system completes all analysis runs **without crashes**, even when datasets contain incomplete or noisy data that must be cleaned or interpolated.
- Repeated runs using the same dataset and configuration produce outputs with no more than **2% variation** in detected fault locations.

6.2.4 Scalability:

Minimum Benchmark Requirement:

- The system processes any bathymetry datasets provided by the client of the **same size** as the largest dataset used during testing, **without memory-related failures**.
- Processing time scales **proportionally with the dataset size**
  - Doubling the number of grid cells in the datasets does not increase runtime by more than **double**.

Target (Desired) Benchmark:

- The system handles datasets at least 2x larger than the largest dataset used during testing.
- Processing time should **scale sub-linearly**, with dataset doubling resulting in **less than 1.5x increase** in runtime.

By meeting these performance benchmarks, the system will provide geologists with a faster, more reliable, and scalable workflow for fault detection compared to existing manual methods.

6.3 Environmental Requirements:

In this section we will describe the environmental constraints/non-functional requirements of the project which arise from the nature of the project itself. These are things that result from our client's use case, the use of geospatial data, and the use of machine learning. They are as follows.

1. The use of the NetCDF data format and its native library

   a. The bathymetrical data will be downloaded through GeoMappApp as NetCDF data. A complex, self-describing data type which is created for multidimensional scientific data and requires a library(NetCDF4 Python) to manage effectively.
   b. This ensure compatibility with our clients existing programs wince most that pertain to this area of study use NetCDF
2. The output as a textfile
   a. Our client has asked that we output the data our program may find as a text file for the sake of simplicity.
   b. This allows our client to easily process and visualize the data with the tools they are familiar with.
3. The program must run on Linux
   a. Our clients primarily rely on Linux for working with their data and thus it is important that we make sure our program uses that same OS for creating our program.
4. The use of Python
   a. Our program must use Python since this language provides strong cohesion for machine learning. Some of the packages that we are using, like Scikit-learn, are only compatible with python. And all of our other packages which are compatible with multiple languages can be used with Python.
   b. This will also make the project easier to maintain down the line.

7. POTENTIAL RISKS

In this section, we will outline the potential risks that could affect the success, reliability, or usability of our project. Because our system is designed to automate fault detection using ML, risks are primarily related to data quality, model accuracy, and user interpretation of results. Each risk includes a description of the likelihood and our planned mitigation strategy.

Risk	Description	Impact	Likelihood	Mitigation Strategy
Data Quality	The input data may contain missing or corrupt values.	This could lead to inaccurate fault detection or program failure.	Medium	We will validate and clean data using PyGMT and NumPy before model input.

Model Accuracy	The model may confuse other natural seafloor features for faults.	This could lead to incorrect results and reduced trust in the model.	High	We will use a hybrid CNN + clustering + Random Forest approach and review the outputs with our clients.
Overfitting	Our model may not generalize to new regions.	This could lead to poor accuracy on unseen data.	Medium	We will train the model with varied datasets and use cross-validations.
Scalability	Large datasets could exceed memory limits.	This could lead to slow processing or crashes.	Medium	We will implement chunked processing and lazy loading.
Interpretability	Users may not be able to understand why certain faults (or non-faults) were identified.	This could lead to a misinterpretation of results.	Low	We will provide a clear output log and confidence indicators over the original data using PyGMT.

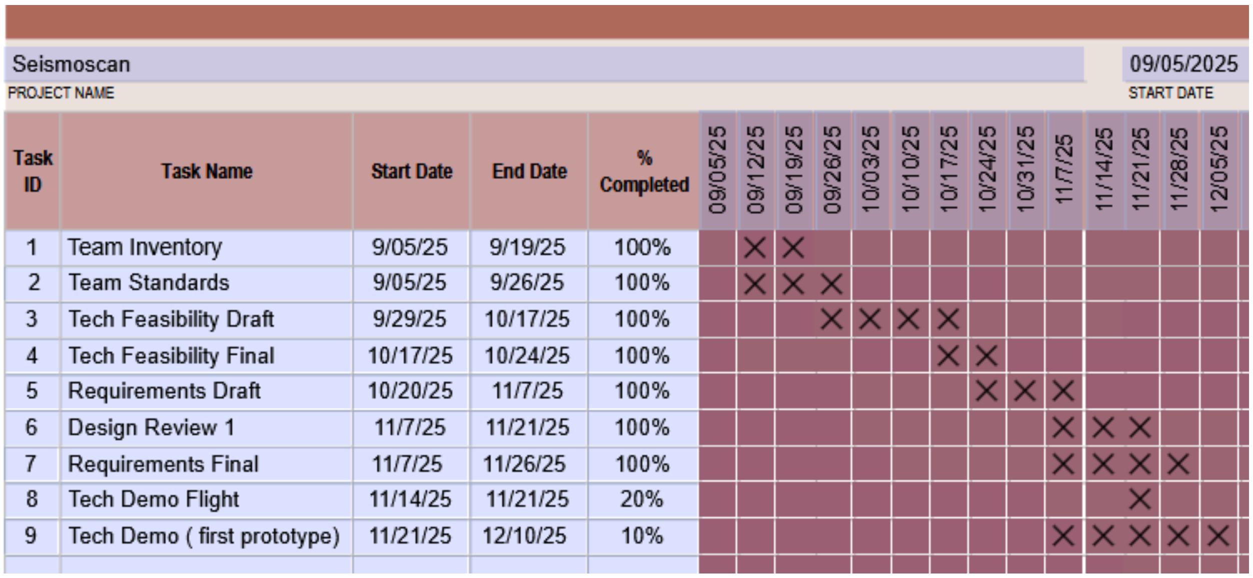
Identifying these risks early on allows us to plan for effective mitigation strategies to be in place for progress in our project. This will also allow us to ensure that our project remains reliable, transparent, accurate, and sustainable for long-term use in geological research as well as maintain client confidence in our project as a research tool.

8. PROJECT PLAN

So far, our plan to create this project has been to discover, research, and start development. For our development process we aim to first create a framework in the form of a working pipeline which integrates all the technologies we want to include working in unison. Then we will develop the individual parts of the pipeline like preprocessing and different machine learning modules to use.

Our milestones will include the various functional requirements.

1. Research technologies, begin assembling pipeline (Command line interface, Ingest NetCDF data,output module)
2. Define our approach to creating data (like slope) so that it may be interpreted by the machine learning modules. Create areas for machine learning modules.
3. Get the machine learning modules to process the data and produce an output that is consistent with the model used.
4. Complete the pipeline (by end of semester one). At this point the program will be able to take an input and produce an output. The output probably won't be very good.
5. Fine tune the machine learning modules to create a better output.



Seismoscan
PROJECT NAME

09/05/2025
START DATE

Task ID	Task Name	Start Date	End Date	% Completed	09/05/25	09/12/25	09/19/25	09/26/25	10/03/25	10/10/25	10/17/25	10/24/25	10/31/25	11/7/25	11/14/25	11/21/25	11/28/25	12/05/25
1	Team Inventory	9/05/25	9/19/25	100%		X	X											
2	Team Standards	9/05/25	9/26/25	100%		X	X	X										
3	Tech Feasibility Draft	9/29/25	10/17/25	100%				X	X	X	X							
4	Tech Feasibility Final	10/17/25	10/24/25	100%							X	X						
5	Requirements Draft	10/20/25	11/7/25	100%								X	X	X				
6	Design Review 1	11/7/25	11/21/25	100%										X	X	X		
7	Requirements Final	11/7/25	11/26/25	100%										X	X	X	X	
8	Tech Demo Flight	11/14/25	11/21/25	20%												X		
9	Tech Demo (first prototype)	11/21/25	12/10/25	10%										X	X	X	X	X

So far we have spent most of our time completing the Tech Feasibility paper and Requirements draft. Just last week we had our first design review which was a poster presentation and a powerpoint presentation. At the start of November we started doing research to build our prototype for our tech demo.

9. CONCLUSION

The project our team is working on is a Python-based machine learning model that's able to accept bathymetry data to record faults in the oceanic tectonic plates. Through this innovation we are able to help our clients who normally spend hours hand drawing these faults. By creating a machine learning model that will be able to automatically identify points as faults within bathymetry data. This will save our client time even if it takes just as long to run as it will allow them to spend their time doing other things as well as advocating with further research on tectonic plates.

In order to do this we will build a command line interface that connects with a machine learning model. This interface will be very simple to allow for quick use. The machine learning model will run and then will output the identified locations into a .txt file as points of latitude and longitude.

We demonstrate in this document the most important requirements of the project. The MVP will require a command line interface, the ability to ingest NetCDF data, process that data using a machine learning model, and to complete the process it will output the latitude and longitude to a .txt. We were also able to show that the efficiency of the program has little importance and that the correctness of the model is the key aspect in this project. Then demonstrating some of the risks that might come with the project, the largest of which being accuracy. Finally we put a plan together to complete this project.

Through this Requirements acquisition process we were able to make strong headway on discovering and deciding the importance of all the features that will be implemented. Some of the most vital pieces of information is the lack of UI needed for this project, the speed performance of this project being irrelevant, and the sense of scale that this project contains using data from all around the world. With respect to development progress has been made from multiple different angles, including the beginning of using different machine learning algorithms, and the implementation of data cleaning is well underway.