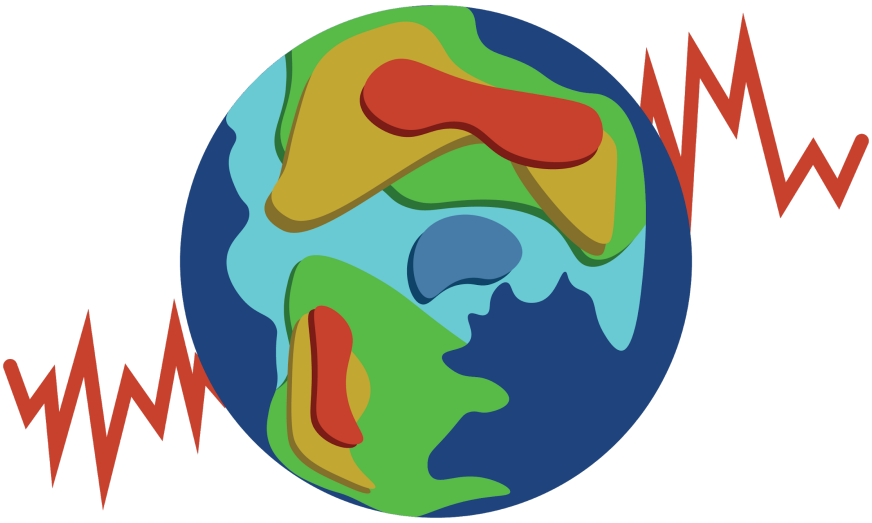
Requirements Specification



Date: 10/30/2020

Team Name: Presto Proxies

Team Members: Melissa Peiffer, Mason Rodgers, Justin Coffey, Colin Taylor

Project Sponsor: Dr. Nicholas McKay

Team Mentor: David Failing

Accepted as the baseline for requirements for the project.

Client Signature: _____ Date: 11/24/20

Team Signature: _Presto Proxies___ Date: ____12/02/20_____

Table of Contents

Introduction

Paleoclimatology is the study of past climate conditions. Specifically, Paleoclimatologists study variations in the paleoclimate in order to identify the causes of climate change. Past climate conditions can be observed by using natural records such as tree rings or ice cores which contain imprints of the Paleoclimate. These imprints are known as proxy records. For example, the isotopes preserved in coral skeletons can be used to measure the temperature of the water in the area where the coral grew. Proxy records can range anywhere from lake sediments to pollen, and provide measurable climate data, such as temperature, precipitation, and air pressure in a specific region. By collecting this data, Paleoclimatologists can create climate reconstructions that accurately model past climate conditions and variations.

Our project is focused on providing a simple way to analyze Paleoclimate data so that researchers can easily determine climate variations and causes. The project will not only be instrumental to the research of Paleoclimatologists, but also for policymakers and educators. Using inferences from the Paleoclimate visualizations, policymakers will be better informed to enact laws that mitigate the effects of climate change. Educators can use the information from the Paleoclimate visualizations to teach future generations ways to protect the natural environment. In order to comprehensively address climate change now and in the future, it is crucial that we first understand past climate variations and their causes.

Problem Statement

The PreSto visualization project is sponsored by Dr. Nicholas McKay and his collaborators at the University of Southern California. Currently, the Paleoclimate Dynamics Laboratory is collecting data from lake sediments and tree rings in Southern Colorado. Using these proxy records, Dr. McKay and his collaborators can create reconstructions to model the ways that dust and drought interact and cause earlier snowmelt in that region. By understanding the cause of rising temperatures and earlier snowmelt in Southern Colorado, concrete actions can be then taken to mitigate the effects of climate change in the area.

Paleoclimate data has been collected from proxy records over the past 30 years. The sample datasets provided to us by Dr. McKay take a minimum 2GB of space each, and this mass amount of information makes it nearly impossible for him and his collaborators to decipher patterns and variations in the Paleoclimate by hand. This presents a major problem for anyone who needs to use the data to inform their decisions or research. Trying to understand what causes climate change without a way to visualize Paleoclimate data and isolate meaningful subsets of information is highly inefficient.

Extensive work has been done to produce algorithms that can build climate reconstructions given Paleoclimate data. Once researchers have collected data from proxy records, PReSto allows administrative users to upload their datasets and select their desired reconstruction algorithm. However, front-end visualizations for these reconstructed datasets require extensive technical and programming expertise to implement. As such, the results of algorithmic computation are highly inaccessible to scientists, policymakers, and educators, who may not necessarily have experience implementing a front end visualization software. These are the end-users who would benefit most from understanding Paleoclimate reconstructions, so it is vital that they have an efficient way to visualize the data provided to them by Paleoclimatologists.

The aforementioned problems that our project seeks to address are summarized below.

- The amount of data makes it difficult to isolate meaningful patterns and subsets.
- Results of algorithmic computation are inaccessible to critical end users without technical knowledge.

In order to make the data more accessible, our project is focused on taking Paleoclimate reconstructions generated from existing algorithms, and providing a front end visualization application for those reconstructions. A more detailed overview of our proposed solution is outlined in the following section.

Solution Vision

Our proposed solution is a visualization application for Paleoclimate data. Users will be able to upload their data, select reconstruction algorithms, and dynamically interact with the resulting visualization. For example, Figure 2 in the appendix shows that a user may upload their dataset, select a line chart as their visualization method, and explore the resulting line chart in both time and space. The end product will allow two main types of data visualization, including geospatial visualization given NetCDF data, and time-series visualization given an index value. Index data is a single value observed over some unit of time, whereas NetCDF data is a set of Paleoclimate data observed in a specific region over a large period of time. Both visualizations will allow users to zoom in on plots and navigate through the visualizations in time and space.

By providing users with a way to visualize their data, users can easily identify meaningful patterns in large datasets and isolate specific data subsets of interest. Isolated data subsets can be exported by the user, along with their selected visualization graph. Ideally, this exportation will include a bibliography containing more detailed information about where the data was obtained, including the specific region and time period associated with that data set. The visualization application will be able to handle sizable inputs, making it extremely easy for users to identify patterns in large datasets. Figure 3 in the appendix shows the proposed architecture for our application. The webserver will take in NetCDF files containing the Paleoclimate data, parse the data using a Python script, plot the data with a Python library, and return the visualizations to the user. The system will provide a visualization front end for algorithmic reconstructions, eliminating the need for end-users to have extensive technical knowledge.

From our proposed solution, we address the aforementioned problems by implementing the following features.

- Users will be able to upload at least 2GB of data to our application and analyze the resulting visualizations in less time than it would take to analyze the data by hand.
- A front end implementation for data visualization will eliminate the need for end-users to have extensive technical knowledge.

The following section will outline our plan for implementing these solutions in more detail.

Project Requirements

Our application will ingest reconstructions form PReSto, processing standard NetCDF files to produce interactive visualizations. Paleoclimatologists will collect their data from proxy

records, or use pre-existing paleoclimate datasets, and pass that information to data stewards. The data stewards are administrative users of PReSto, and will upload the datasets onto the server. From there, an end user, such as a researcher, educator, or policy maker, will be given a set of options. They may choose a specific dataset from a drop down menu on the front end of our visualization software. From there, the software will determine which algorithms are appropriate for that type of dataset, and provide the user with another set of options for their desired reconstruction algorithm. These reconstruction algorithms will be stored as static files on the webserver for the visualization application.

Once a user has selected a dataset and reconstruction algorithm, they may choose from a set of specific visualization types. The first type of visualization is a time series visualization, allowing the user to see the data in both time and space, and providing shading to represent uncertainty. The second type of visualization is a geospatial visualization, allowing the user to see the data as a map, representing changes and temperature and precipitation as color gridded overlays on the resulting map. For both types of visualizations will allow users to drag and select areas of interest as well as enter in more specific parameters for time and region.

Once the visualizations have been produced, and the user has isolated a pattern of interest from their data, they will be able to export the visualization and associated data subset. Finally, as Dr. McKay, his collaborators, and students will be maintaining the website after the final product is delivered, the application will have administrative capabilities. In the following sections, we discuss each project requirement in more detail.

Functional Requirements

1. Processing NetCDF files

The data used for all visualizations will come in the form of NetCDF files. The files will be provided by the client and will be stored on the server-side of the application. Because all of the paleoclimate data will be stored as a NetCDF file, we must ensure that our web application can accommodate files in this format.

a. Automated ingestion and processing of standardized NetCDF files

Early on in the development of this application, we will be using sample NetCDF files saved on the server to perform testing. For a fully implemented visualization application, we are hoping to make all files in PReSto available for use. Because PReSto will always be adding and changing the data it has available, having some way to automatically update that data would save an immense amount of time and effort for future maintainers. This will be instrumental in future-proofing our project for our client and any other maintainers.

Necessary Functions

Check Dataset is New

- A Boolean (True/False) function to identify whether a dataset is new. This will be a helper function for all following functions.

Get Available NetCDF Files

- If an administrative user adds in a new NetCDF file to the server, or removes an existing NetCDF file from the server, the application should check which files have been provided and update the list of possible datasets.

Process NetCDF Files

- Any new NetCDF file added to the server by an administrative user should be processed so that the application can use it in its computations.

2. Interactive web interface

Our client wants the web application to be accessible to anyone, regardless of technical knowledge.

a. Simple user interface

This web application is meant to be used by anyone. Because of this, we must provide an easy and intuitive user interface. This means that we will keep the user interface as simple as possible without losing any important utilities. *Figure 2* in the appendix shows a mockup for the user interface. The interface includes areas for the data, algorithm, and visualization type to be selected, as well as criteria to adjust the resulting figure. It also includes several options for figure exportation.

Necessary Functions

Select Dataset

- Provide a drop down menu of available datasets for the user to select.

Select Visualization Type

- Provide a drop down menu of visualization types for the user to select. This includes geospatial and time series visualization options.

Select Reconstruction Algorithm

- Application selects appropriate reconstruction algorithms for given dataset, and provides the resulting algorithms to the user in a drop down menu of options.

Select Figure Exportation Type

- Provide three buttons for the user to select from, showing options for PDF, JPEG, or PNG exportation types. Application checks first if a visualization has been generated, and throws an error if it has not yet been generated. Otherwise, the application takes the visualization and exports it based on the user selection.

b. Administrative privileges

After the final product is delivered, the web application needs to be accessible for maintainers. In order to allow changes to be made to the system, we will provide administrative privileges to the client and future maintainers.

Necessary Functions

Upload NetCDF File

- Allow the administrator to add new NetCDF files to the server.

Delete NetCDF File

- Allow the administrator to delete existing NetCDF files from the server.

Provide NetCDF data information

- This is a long term goal for the bibliography. Administrators should be able to provide additional information about where NetCDF datasets were obtained, such as the specific study and proxy record type.

Upload Reconstruction Algorithms

- Allow the administrator to add new reconstruction algorithms to the server.

Delete Reconstruction Algorithm

- Allow the administrator to delete existing reconstruction algorithms from the server.

3. Creation of Visualizations

Our web application will take in the requested NetCDF files and use the data to create visualizations. These will be customizable to some extent, to better show off the desired data.

a. Selection of visualization type

Our program must provide two main types of data visualization. The first is a geospatial map. The second is a time series plot. If a user selects a time series visualization, they may also choose a more specific type of plot, such as a line graph or a boxplot.

Necessary Functions

Get Visualization Type

- Get the corresponding visualization type given user input.

Generate Visualization

- Based on the selected visualization type , using a visualization library, the application should generate the appropriate visualization based on the dataset.

b. Color-gridded overlays

Some of the visualizations requested by the client will be geospatial maps. The geospatial maps will include color-gridded overlays of the map to represent temperature or precipitation changes. Using these overlays will be helpful for analyzing region-specific data.

Needed Functions

Calculate Average Temperature

- Using the NetCDF data, the application should calculate the average temperature per region.

Calculate Average Precipitation Levels

- Using the NetCDF data, the application should calculate the average precipitation levels per region.

Create Overlays

- Using the results of the above functions, the application should provide a color coded overlay for the visualization, using specific colors and a graph legend to represent the calculated averages.

c. Dynamic changes through space and time

Visualizations should be adjustable to some degree if the user wishes to isolate a more specific area. This should be done intuitively, either by interacting with the visualization itself or by adjusting the parameters of the figure; Allowing users to drag and select regions of interest and see the resulting changes.

Needed Functions

Change Data Parameters

- Provide users with an option to edit the time period or region for their data, and generate a visualization based on the changed parameters.

Zoom In

- Allows users to zoom in on the visualization.

Drag Visualization

- Allows users to click on the visualization and drag it.

d. Intuitive and creative visualizations of uncertainty

Reconstruction algorithms may be run several times with varying inputs to simulate uncertainty. The degree of uncertainty should be represented as shading on the visualization graph, with darker areas representing more confidence, and lighter areas representing less confidence.

Needed Functions

Get Uncertainty

- Uncertainty is calculated by the reconstruction algorithms. The application should be able to get this calculated value from the selected algorithm.

Show Uncertainty

- Once the uncertainty has been isolated, display it as a shaded value on the visualization.

4. Exportation of Visualizations

Visualizations should be accessible to users. In order to facilitate this, users need the ability to export any visualization they create, with a variety of options for exportation

a. Flexible data export

Visualizations should include a variety of options for exportation. These will likely include file types such as JPEG, PNG, and PDF. *Figure 2* in the appendix shows a section of the front end used to export visualizations.

Needed Functions

Get Exportation Type

- Isolate the specific exportation type selected by the user.

Check for Visualization

- Check to see if a visualization has been generated before attempting to export the figure.

Convert Visualization

- If a visualization exists, convert it to the selected file type

Export Visualization

- Once visualization has been converted to the proper file type, application should allow users to export it.

b. Bibliography

Users should be able to track information about where the data was obtained. Specifically, the bibliography should be a separate document that contains information about the study where the proxy data was collected. Currently, this is a long term goal.

Needed Functions

Get NetCDF Data information

- Get the additional NetCDF data information provided by administrators and allow users to export this in a bibliography, along with their visualization.

Non-Functional Requirements

1. Take in NetCDF files

The data used for all visualizations will come in the form of NetCDF files. These will be provided by the client and will be stored server-side. Because all of the paleoclimate data will be NetCDF, we must ensure that our web application will be compatible with that.

a. **NetCDF File Load Time**

NetCDF files will be loaded into memory into local memory in under 500 ms (estimate) for a standard 2 GB NetCDF file.

2. Create Visualizations

Our web application will take in the requested NetCDF files and use the data to create visualizations. These will be customizable to some extent, to better show off the desired data.

a. **Visualization Generation Time**

Visualization speed is an important factor when creating a web application. Once the required data is loaded into memory, completion of visualization will take from 1 to 10 seconds (estimate) depending on both the visualization type and the size of the dataset chosen by the user.

b. **Visualization Keys**

Users will be provided with a key for their data. This will allow them to interpret the resulting visualizations, ensuring that the data is easy to analyze and understand.For time series visualizations, the keys will include axis labels such as time, and year. Time series visualizations will also include a title to show the specific value being displayed, such as mean temperature, precipitation, etc. Geospatial visualizations will include a key showing what the colors on the color gridded overlay represent. For example, red might represent a higher mean temperature, whereas blue might represent a lower mean temperature in a region.

3. Export Visualization

We want these visualizations to be usable by many people in a variety of ways. In order to facilitate this, we want the user to have the ability to export any visualization they create, with a few export options.

a. **Flexible Data Export**

Users will be able to export to a specific set of at least 3 different common image formats. Visualizations will be under 10 MB in all file formats.

Environmental Requirements

1. Server-side storage

For the project, we are going to have the files stored server-side. Currently, we do not know the amount of storage we will have, or if we can increase the storage if requested. Because the available server-side storage is unknown and limited, we need to have an alternative to the provided server for storing static files.

Potential Risks

The reconstruction algorithms used by the system will be stored as static files on the webserver. However, we do not know the exact amount of space available on the webserver where we are hosting our website. A potential risk that we may face is attempting to store algorithm reconstruction files that exceed the available space on the webserver. This risk would have dire consequences for the project. Without the reconstruction files, a critical piece in our system architecture would be missing, and the website would be unable to produce visualizations. Based on the size of the files we are using in our prototypes, we anticipate this risk to have a high likelihood of occurrence. If this becomes the case, it could change implementation details that would incur extra time to correct. In order to address this issue, we may need to store the reconstruction algorithms on a separate webserver than the one provided.

Another potential risk that may occur is receiving incorrect user input for the selected dataset, reconstruction algorithm, or reconstruction algorithm parameters. This risk would have moderate consequences for the project. Most likely, an error would occur if the user gave an incorrect input to the system, and as a result, they would be unable to create a visualization. However, this would not break the program, and users would be able to try again with a correct input. We anticipate this risk to have a low likelihood of occurrence. In order to address this issue, we will provide built-in error checking for user input, and may eliminate the need for user input. For example, rather than having a user type in the filename for the desired algorithm, we may include a drop-down menu of options for the user to choose from.

Finally, more reconstruction algorithms may be added to the server after the final product is delivered. Potential risk might occur if the program is not built to accommodate those additional algorithms. This would drastically limit the program capabilities, as it would not allow for flexibility of use. We anticipate this risk to have a high likelihood of occurrence, as many reconstruction algorithms exist and we are currently only developing visualizations for a small subset of those algorithms. In order to address this issue, we must develop our program in such a way that it can accommodate any provided reconstruction algorithm.

We do not expect these risks to affect our development schedule. Our evaluation of potential risks provides us with an understanding of areas in our project where we will need to

add additional countermeasures for errors. In the following section, we outline our project schedule for the remainder of the semester.

Project Plan

Figure 4 in the appendix outlines our team's tentative development plan for the visualization software.We expect the visualization scripts and back end development to take the majority of our time, while we expect front end development and final changes to take less time. We have allocated our time to reflect this. Starting in January, our group will begin developing the visualization scripts, allowing for a month and a half of development time. We also anticipate taking a month and a half for the back end development. Once these two aspects of the project have been completed, we will begin development on the front end user interface, allowing for a month of development time. Finally, we will leave the last month of the semester open to add in any final changes to the project and perform tests. If time permits, we will use this month to develop one of the 'reach' goals for the project, which is a bibliography telling the user where the proxy data was obtained. Overall, we anticipate that the project will take four months to complete.

Conclusion

Climate reconstruction algorithms exist to model the past climate. However, to make these reconstructions more accessible to critical end-users, a front end visualization application is essential. Our project will provide this missing piece, giving users a convenient way to analyze patterns in their data. Understanding past climate variations is essential to addressing current and future climate change. Our project is focused on providing a way for researchers, educators, and policymakers to find patterns of climate variation and identify causes of climate change. Using this information, efforts to mitigate climate change can be backed up by understandable visualizations of data. Having credible information about climate change can help our society take concrete actions to prevent worsening climate conditions.

The project will take in NetCDF files containing proxy record data, allow users to select a reconstruction algorithm, and generate interactive visualizations from their data. The visualizations will allow users to move through time and space and isolate meaningful data subsets for exportation. The program will also have administrative capabilities, so that future maintainers will be able to make changes to the software as needed after the final product has been delivered.

A potential issue we foresee is running out of space on the provided webserver, as the static algorithm files stored there may be quite large. We will overcome this issue by hosting the static algorithm files on a different server altogether. We must also include error checking and predetermined choices to prevent incorrect user inputs. Finally, we will need to allow administrative users to provide additional reconstruction algorithms to the server.

Overall, our project is going according to schedule, and we have outlined our plans for future development. We have started to prototype the program and will be presenting a technical demonstration of the product in the near future. With our schedule and current progress, we are confident that we can deliver a quality product by the end of the next semester.

Appendix

Figure 1: Conceptual design of PReSto. This project focuses on the web front end (grey box in the middle)

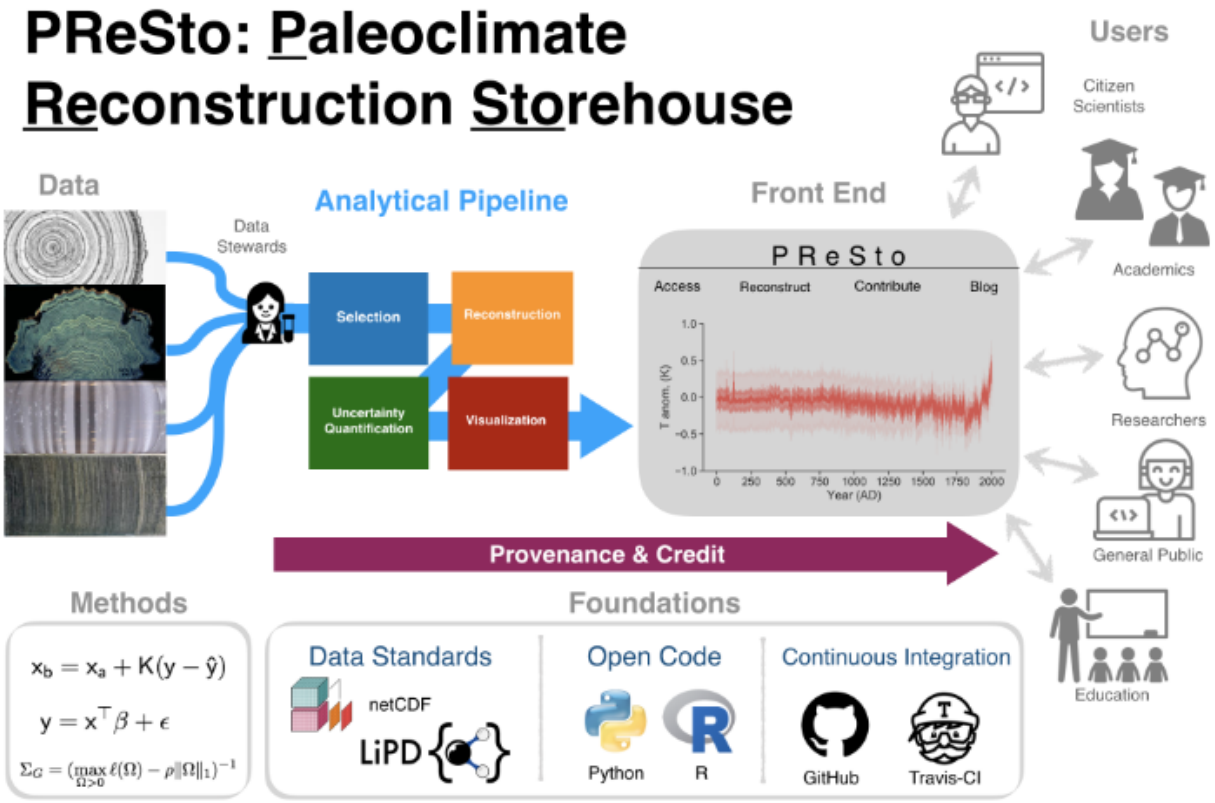


Figure 2: Example design for the Front End, using Matplotlib for plotting

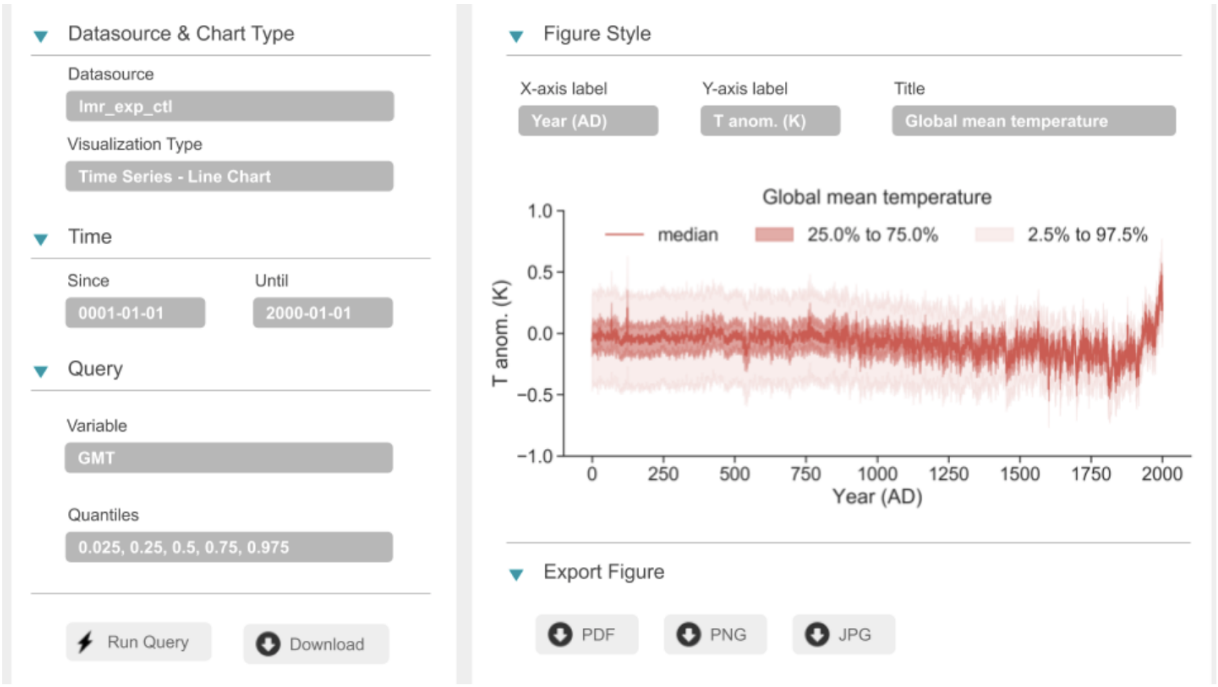


Figure 3: Example design for system architecture

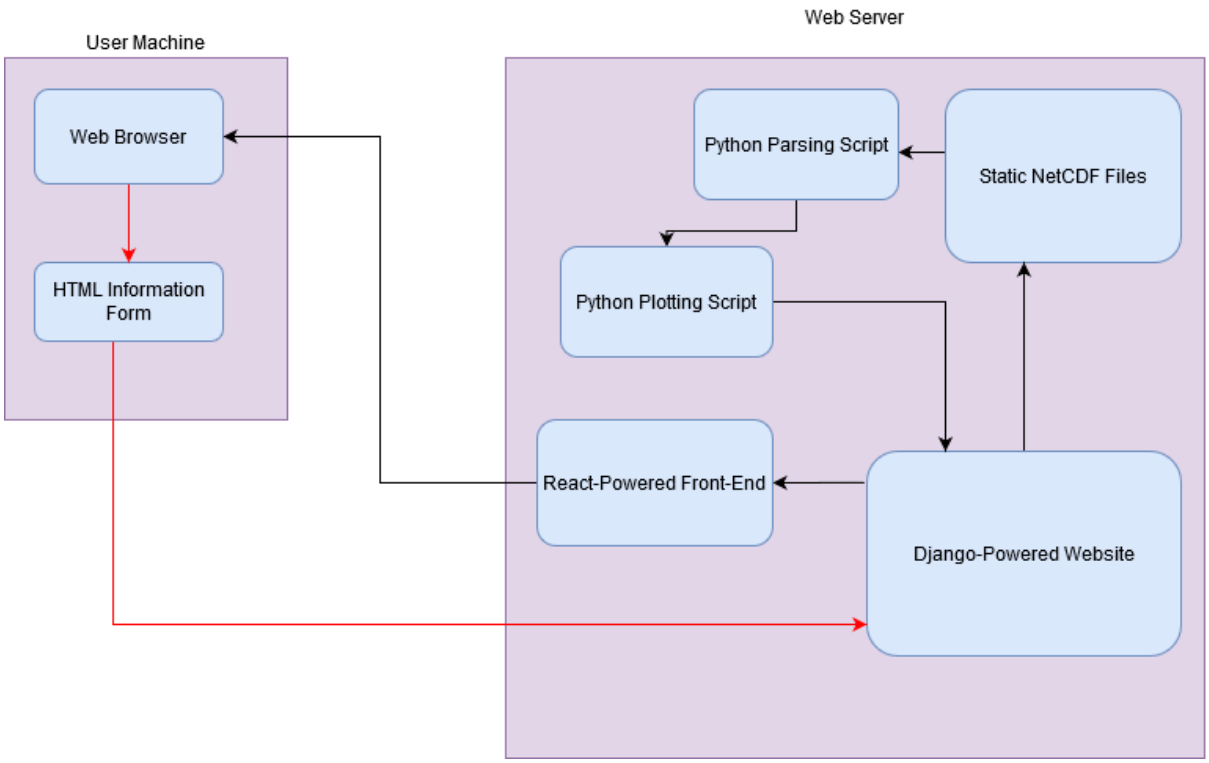
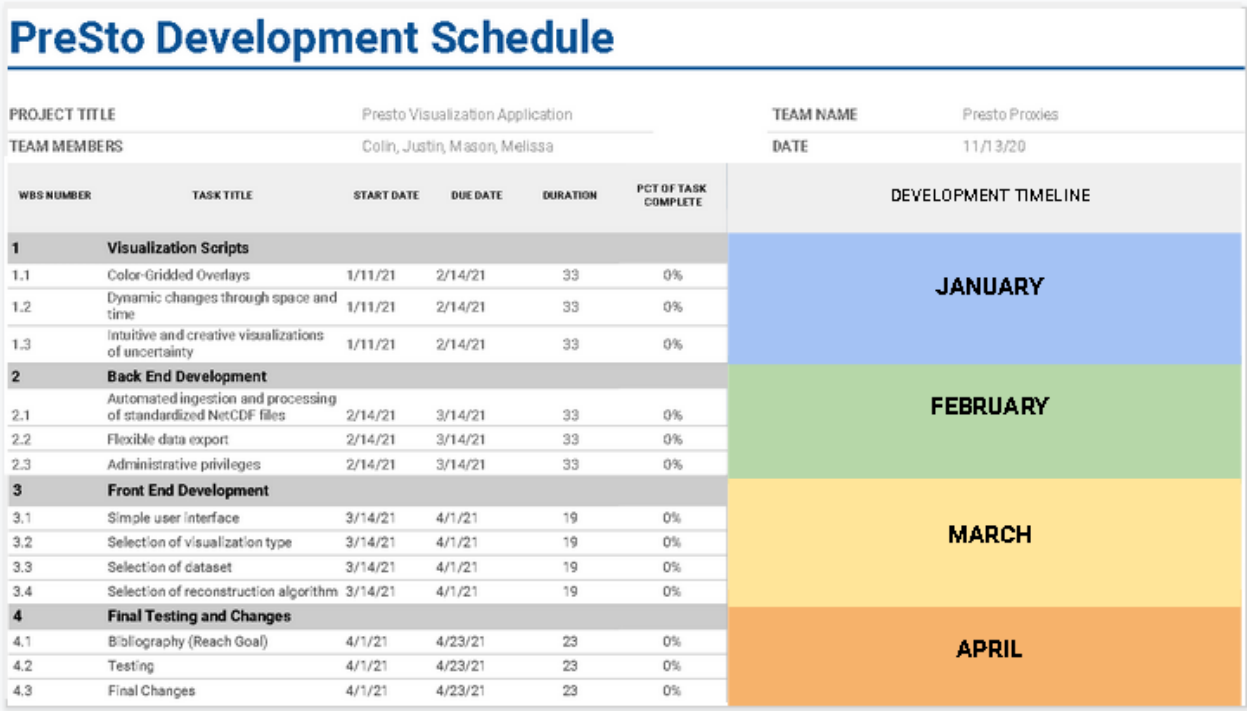


Figure 4: Tentative Project Development Schedule



PreSto Development Schedule

PROJECT TITLE	Presto Visualization Application	TEAM NAME	Presto Proxies
TEAM MEMBERS	Colin, Justin, Mason, Melissa	DATE	11/13/20

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE	DEVELOPMENT TIMELINE
1	**Visualization Scripts**					JANUARY
1.1	Color-Gridded Overlays	1/11/21	2/14/21	33	0%	
1.2	Dynamic changes through space and time	1/11/21	2/14/21	33	0%	
1.3	Intuitive and creative visualizations of uncertainty	1/11/21	2/14/21	33	0%	
2	**Back End Development**					FEBRUARY
2.1	Automated ingestion and processing of standardized NetCDF files	2/14/21	3/14/21	33	0%	
2.2	Flexible data export	2/14/21	3/14/21	33	0%	
2.3	Administrative privileges	2/14/21	3/14/21	33	0%	
3	**Front End Development**					MARCH
3.1	Simple user interface	3/14/21	4/1/21	19	0%	
3.2	Selection of visualization type	3/14/21	4/1/21	19	0%	
3.3	Selection of dataset	3/14/21	4/1/21	19	0%	
3.4	Selection of reconstruction algorithm	3/14/21	4/1/21	19	0%	
4	**Final Testing and Changes**					APRIL
4.1	Bibliography (Reach Goal)	4/1/21	4/23/21	23	0%	
4.2	Testing	4/1/21	4/23/21	23	0%	
4.3	Final Changes	4/1/21	4/23/21	23	0%	

Additional Resources

Heiser, Christopher, et al. " Linked Paleo Data ." *LiPD*, Paleoclimate Dynamic Laboratory, lipd.net/

National Geographic Society. "Paleoclimatology." *National Geographic Society*, National Geographic, 9 Sept. 2019, www.nationalgeographic.org/encyclopedia/paleoclimatology-RL/.

"Paleoclimate Dynamics Laboratory." *Paleoclimate Dynamics Laboratory - Northern Arizona University*, www.cefns.nau.edu/~npm4/index.html.

"What Are 'Proxy' Data?" *National Climatic Data Center*, National Oceanic and Atmospheric Administration, www.ncdc.noaa.gov/news/what-are-proxy-data.

Glossary

Climate - The long term weather patterns observed over many years in a particular region.

Django - A back end framework implemented in Python.

NetCDF - Network common data form, a data format for scientific data.

NetCDF4 - A Python library for manipulating NetCDF data.

Paleoclimate - The prehistoric climate.

Proxy Record - A natural record of past climate conditions, such as an ice core or coral skeleton.

React.js - A front end framework implemented in Javascript that provides templates for the User Interface of a project.