# Final Report

May 3, 2020

Version 1.0

**Project Sponsors**

Doctor Geoffrey Roest

Doctor Kevin Gurney

**Team Members**

Kiley Jacobs (Team Leader)

Tung Nguyen

Zihang Shen

Yisheng Wang

**Team Faculty Mentor**

Scooter Nowak

## Table of Contents

## 1. Introduction

The Earth is warming at an alarming rate, and the growth of CO2 emissions is a major reason for this change. Many countries have signed agreements to reduce CO2 emissions, and many cities in the United States, like Los Angeles, are taking the lead in reducing CO2 emissions. However, actions like diplomatic agreements are not enough. Without an easy to use way for everyday people to see environmental changes, people cannot observe and understand the changes of regional CO2 emissions. Also, people don't know whether their actions really mitigate global climate change or not. So, our project is to create a Web Application that can give people a very clear visual map that shows how much CO2 emissions are increasing, as well as being able to see the CO2 emissions level for their own hometown.

The project sponsors are Geoffrey Roest, Postdoctoral researcher in SICCS (School of Informatics, Computing and Cyber Systems) at NAU ( Northern Arizona University), and Kevin Gurney, Professor in SICCS at NAU. Professor Kevin Gurney has recent research in topics in carbon cycle science, climate science, and climate science policy, and is working on a project involving simulation and quantification of U.S. CO2 emissions, the linkages between terrestrial carbon exchange and climate variability, and the impacts of deforestation on climate. He also has worked extensively on climate policy and has been involved for over 25 years with the United Nations Climate Change Framework Convention and the Kyoto Protocol.

For this project they have collected large amounts of data about CO2 emissions in the United States, and have been able to create a map over the country that displays this data. The problems that they are facing right now are dissemination and accessibility. Since the data is only available in technical formats, you need specific software to open and analyze them; the general public doesn't have sufficient knowledge or skills to use most of this software. So, our primary goal is to build an easily accessible interface that allows everyday people to understand and explore CO2 emission findings. Our solution

will be a Web application that can show the CO2 emissions in an interactive graphical interface. Users can access our website and see a map of America displaying relative CO2 emissions. This map will be dynamic, allowing users to change things like color and transparency, as well as view information about specific areas on the map.

In this document, we will be going over the entire process and final product that was created as a result of our Capstone experience. In the following sections, the Process Overview, Requirements, Architecture and Implementation, Testing, Project Timeline, and Future Work will be covered in detail.

## 2. Process overview

At the beginning of the year, our team assigned roles to each team member preliminarily, which have evolved and been reassigned as our individual strengths have become clear. Initially, the roles assigned were:

• Team Leader: the team leader will run meetings, follow-up on team deadlines and productivity, schedule meetings with the client, and ensure each team member is represented and able to provide opinions.

• Customer Communicator: the customer communicator will be the primary point of contact for the clients and be available to the client as appropriate.

•Proofreader: the proofreader will review all team documents and deliverables for cohesion, formatting, and professionalism before submission. They will also be responsible for ensuring documents are printed and submitted.

• Release Manager: the release manager will coordinate version control. This includes ensuring readability and cohesion of documentation.

• Coder: programming for each function and combining the functions together.

For each component, someone would update and create files in Google drive, and also upload them to Slack, our communication channel. At times, it was possible we could each be working on different components of the same file, or perhaps formatting a file

while another was working on a content of it, complicating merges. Software components were not developed in a specific order, except for what was intuitive for us. Each week we would assign tasks for the next, based on what we thought was appropriate for the components and tasks at hand. Some modules needed to be developed linearly, i.e., we had to have a map that needed displayed before we could overlay anything over it.

# 3. Requirements

For an interactive map-based website, users' experience is the most important aspect. Our website must be a highly functional website that is easy to use with no frustration on the user's end. This will allow us to create several different functions to allow the user to interact and customize the website's map in different ways.

## 3.1 Functional Requirements

Functional Requirements needs to layout all of the functions that the application will have, as well as how various users can use our application.

### 3.1.1 Our Product will Upload Raster Data Formats

Our clients already possess, and will provide us with, sufficient CO2 emissions data for America. Accessing this data using our maps will be one of the main requirements. Our clients have been helped previously by google programmers to create an interactive map using GIS, but the lack of maintenance has caused the collapse of that interactive map. Most of the data the clients currently have is GIS raster data. This raster data was given to us in the form of Geotiff files which is an image file similar to that of a PNG or JPEG. We need to upload this data into the Mapbox, which is the map API we have chosen to utilize for our project.

### 3.1.2 Show Data on Map

After we upload the data into datasets and/or tilesets, the next step is to use this data and show it on our map. We will need a Mapbox access token to use any of

Mapbox's tools, APIs, or SDKs, as Mapbox uses access tokens to associate the account with the requests to Mapbox API resources. We can find all the access tokens, create new ones, or delete existing ones on the Access tokens page. We can also use the Mapbox Tokens API to programmatically create, update, and delete access tokens.

#### 3.1.3 Our Product will Provide User interaction in map

We are using maps to display the United States' CO2 emissions, making it easier for ordinary people to understand the changes in CO2 emissions throughout different regions at varying times. Through various user interactions, users can easily gather the information they want about the current state of CO2 emissions in the country.

##### 3.1.3.1 Our Product will Change Map By Year

This function will allow the user to change the year to view CO2 emissions from different times, spanning across several years. This feature makes it more intuitive for users to truly see how CO2 emissions in American are rising or falling over time.

##### 3.1.3.2 Our Product will Change Layer by Emission Sources

Since on our map we expect to show different CO2 emission source layers, such as vehicle exhaust emissions, emissions from factory production, etc., our map will allow users to choose a layer of the map they are looking at at one time.. Users can use this function to switch sources of CO2 emissions that they are interested in, so they can have a better understanding of that particular set of data rather than being overwhelmed by all of them.

##### 3.1.3.3 Our Product will Provide the Search Location Function

Allow users to search for a location, making it easier for users to find the specific place they want to see on the map. Searching the location can be searched by

address or keyword, and Mapbox Studio comes with such a search function already implemented, which entails using Geolocate to locate the user and then track their current location on the map using GeolocateControl.

#### 3.1.3.4 Our Product Will Show Information for Each State

When the user clicks the black dot on the map, it will display the information about that given state, such as the total CO2 emissions in this area, different CO2 emissions from different sources, or the emissions per capita. It will give the user a summary of the local basic information to allow for more context.

## 3.2 Non-functional Requirements

Non-functional requirements are stated as a requirement that can be used to judge the system as a whole instead of specific behaviors. Our non-functional requirements ensure that our system will remain operative and useful to our sponsor and end-users. The three main non-functional requirements that we have obtained are response times of data within one minute, reliability of the application, and accessibility.

### 3.2.1 Quick response from our application to display the necessary data and graphics to our end-users.

We found this non-functional requirement as a core part of the whole application because, if the data is not displayed to end-users in a timely manner, then they may become uninterested and leave the site. To solve this, we have to reduce the amount of layers loaded for each session.

### 3.2.2 Reliability

Given the amount of expected traffic and simultaneous use of the map application, it is extremely important that the system is always accessible to users at all times.

### 3.2.3 The System is easily modified by non-members of our team.

Code will include blocks of comments that are deciphered by non-technical users in less than 10 seconds and able to be modified by minimally experienced programmers. There will be leeway given in areas the client has expressed the intention to expand upon, and comments indicating this.

## 4. Architecture and Implementation

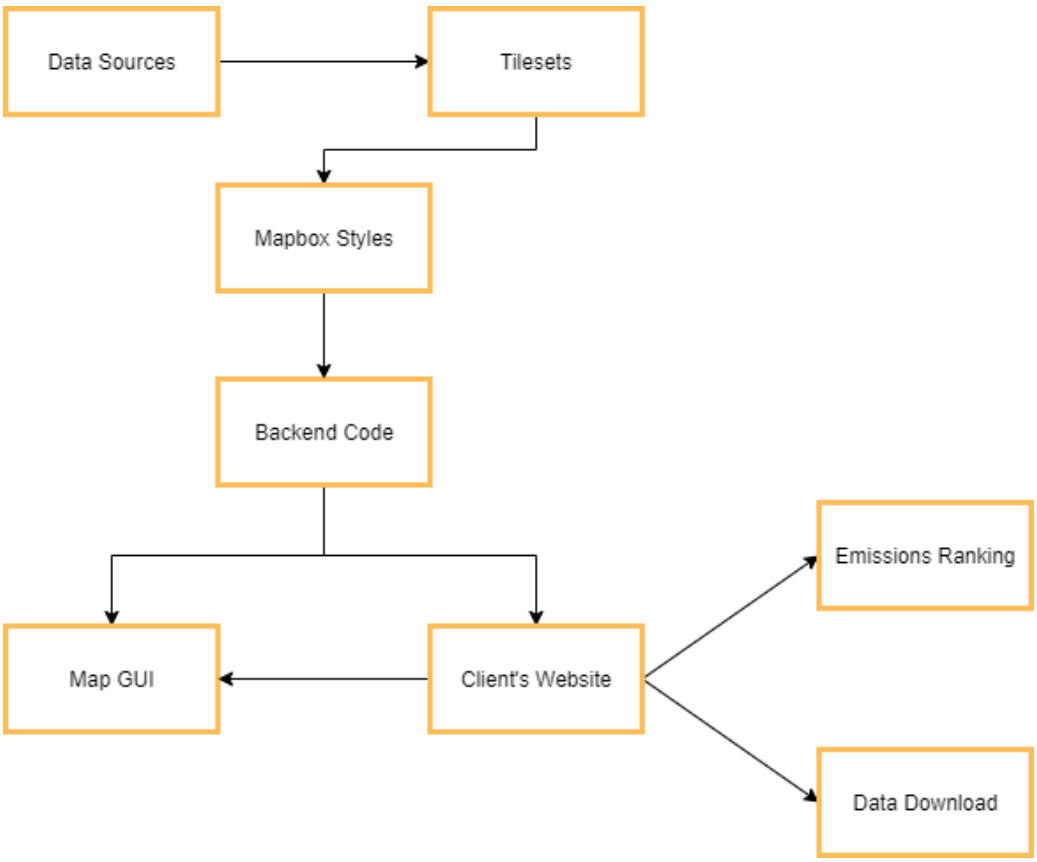
Our primary goal is to build an easily accessible interface that allows everyday people to understand and explore CO2 emission findings. The solution our team has created for our sponsor is an interactive map that can show the CO2 emissions in an interactive graphical interface. Users can access our map and see a map displaying relative CO2 emissions.

We have selected several tools that help facilitate our design. Since we are building a map application, we have chosen to use the Mapbox framework. Mapbox is free, easily integratable, and a JavaScript-based framework. Mapbox allows tileset rendering, and it can handle datasets far above interactive maps that need to load files like geoJSONs directly. Mapbox GL JS, which is mapbox's tool for backend development, also avoids repeated looping, a common issue in interactive maps, by using rendered layers and Mapbox GL JS functions to access specific features.

An important part of a software system is the software development environment, as this has influence on the design, build, and testing. Especially for complex systems, it is important that this has been set up correctly to maintain and guarantee productivity and quality. The architecture of this product is to have users access the webpage on the client's already created website that displays the interactive map. The map is built using Mapbox and allows users to select different layers of the map, along with a number of other features. The user can also see the emission ranking and download data on the webpage.

A general overview of the system flow is depicted in Figure 1. It shows each of the components that make up our project.

*Figure 1: Architectural diagram*




## 4.1 Data Sources

Our data is mainly Geotiff images (raster data), which are data files compatible with Mapbox, allowing us to use backend code to manipulate the way the data is displayed. We can upload up to 10 gb to Mapbox. This data will be uploaded to tilesets to create image layers in Mapbox.

### 4.2 Tilesets

Tilesets are a collection of raster and vector data broken up into a uniform grid of square tiles. It is an optimized way to save and transport data by splitting it into tiles. Mapbox relies on tilesets to keep the maps fast and efficient. Mapbox style will take generated tiles to create a visual map.

### 4.3 Mapbox Styles

Mapbox styles is a document that defines the visual appearance of a map, such as what data to draw, the order to draw it in, and how to style the data when drawing it. We use mapbox style to display the CO2 emissions according to each year. Mapbox style has root properties that provide important descriptive information related to our map.

### 4.4 Backend Code

We use Mapbox GL JS, a Javascript library, to render an interactive map from Mapbox styles. Javascript, HTML, and CSS are used to display emission ranking and data download as well.

### 4.5 Map GUI

The map GUI module is responsible for presenting the map to the user. This component provides the user access to all the functionality of our map. The map interface consists of functions written with Mapbox GL GS.

### 4.6 Client's Website

We will implement our web pages into our client's website. There are three web pages; the map, emissions ranking, and data download. Each page will have their own GUI, with the map being the bulk of our project as it is the most dynamic.

### 4.7 Emission Ranking

This page will display a graph of each kind of emissions as they've changed overtime.

### 4.8 Data Download

The web page has a list of data files sorted by years with a description for users to download. The data will be in the form of csv files and stored on our client's domain.

## 5. Testing

The testing is based on unit testing, integration testing, and usability testing.

### 5.1 Unit testing

Unit testing is mainly used to test each single component and that all the code of the functional modules can perfectly represent the functions in different situations that meet customer requirements. This is to ensure that when all the components are combined, the system will not collapse due to a single component or a single functional module error.

We had testing of all the user interaction on the map, all working well without error.

### 5.2 Integration testing

Another important factor of assuring product quality and integrity is to conduct integration testing. This makes sure that all of the systems, the map framework and the datasets, are all working properly. We had put all the user interaction functions together in our map and tested it on our team website servers. All functions still work well without error. The ranking page shows images well, and the download page's download function is also working well.

### 5.3 Usability testing

Usability testing is about making sure users have the desired experience, by making sure all our functions work during front end interaction as well. Our usability testing will be more straightforward, as our GUI will consist of a number of different buttons and menus for users to interact with. Testing will consist of making sure all these tools work

properly, as well as trying to do things the users shouldn't be able to in order to make sure the functions handle those end cases properly as well. All of the following tests will be carried out by ourselves, as well as asking average users to test if all these things work properly.

During this testing, we got some useful feedback. First one is the initial load time was a little bit long, under the normal internet speed, it took 4 to 5 second. But most testers feel it's not too hard to accept. So, this issue can be improved but not prioritized. The second one is in a different source layer, the same color will represent different values. To solve this problem, we develop a color label function, which shows the different color labels during a layer change. Third one is our sources layer button will not change the color after the user clicks it. Sometimes people may forget the source they are looking for right now, but the label will change to say the name of the layer being shown. This is also not a priority issue, but a good way to optimize our final product.

## 6. Project Timeline

Over the past year, we have focused on gathering requirements from our sponsor and developing our product. We have requirements gathering and development into one to align with the capstone schedule.

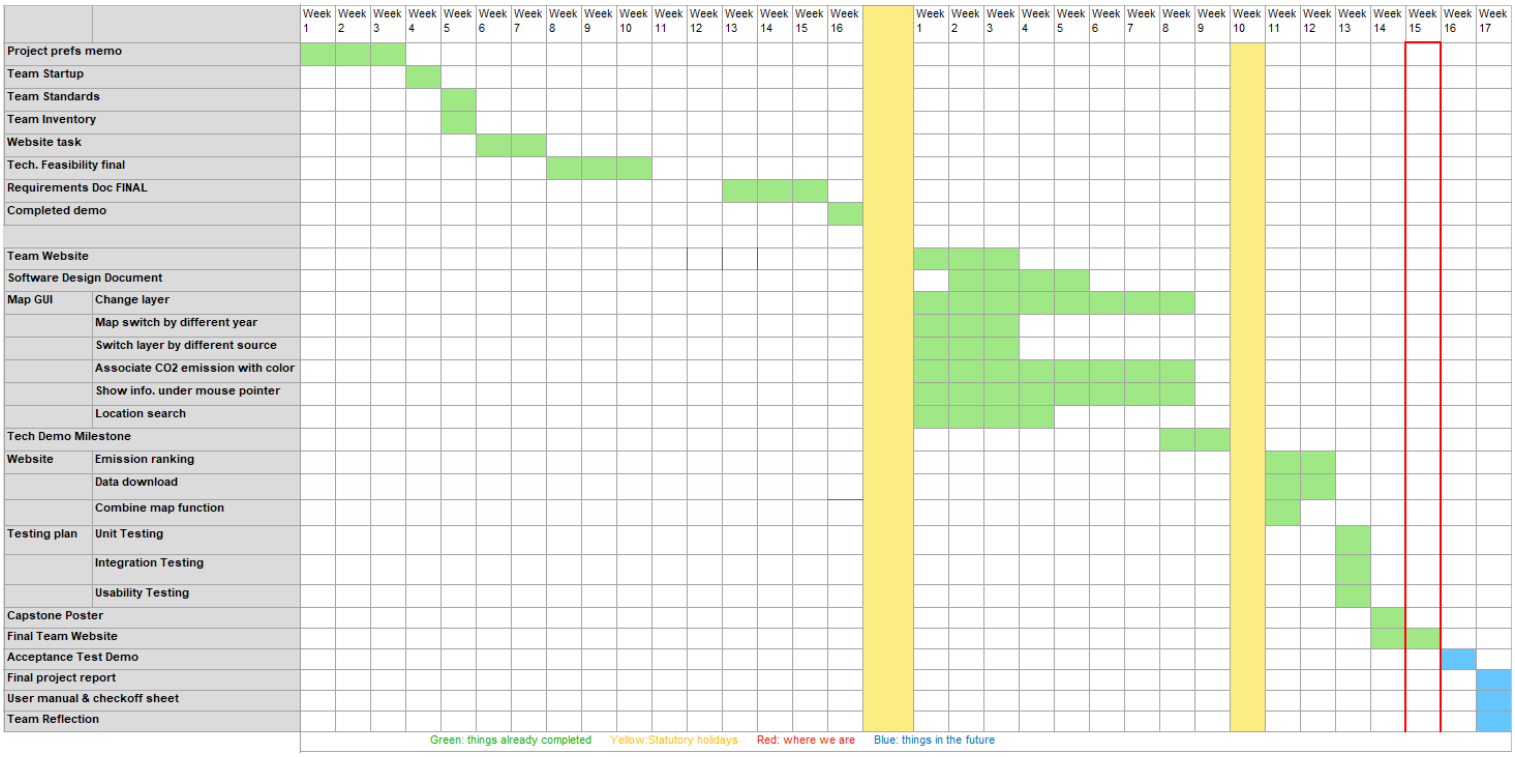



*Figure 2: Final schedule*

When we designed our software, we would describe our technologies we would use, and implement the technologies into a functioning prototype. The gathered frameworks during this time would be changed at the beginning of the second semester. Overall, the schedule outlined our requirements, laid out the initial design and technologies of our product, and ended with a functional prototype that highlights our requirements, design, and proof of concept.

## 7. Future Work

In the future, we can use the Mapbox SDKs for IOS and Android to provide the user interaction map on mobile devices like smartphones. Since these tools are also under Mapbox, it should have some similarity with Mapbox GL JS we used for the web application, so it should not be too hard. We also can improve the map with more detailed data, like from year to months. Or we can add more data to expand the timeline, so that users can see the CO2 emission changes over a longer time.

## 8. Conclusion

In order to let people see CO2 emission in their hometown as well as the whole America, under the sponsorship of researcher Geoffrey Roest and Professor Kevin Gurney, we have come up with a project to create an interactive map, each different part plays an important role in this interaction, and we planned time for us to develop each module and test them as well. We had weekly meetings with our clients where we initially got familiar with their businesses and workflow to help orient our solution to solve their problems.

The overall project in the past year went relatively smooth. Most of the assignments and tasks were split evenly between the four of us and were done on time. Our weekly, iterative process of requirements and development has made our product progress at an optimal pace. The organization of the team, tasks, and weekly meetings made for a smooth experience both for the project and the class. Overall, the Capstone class and the project flow were great and we have succeeded in creating an optimal product for our client over the year.

## 9. Glossary

**API:** Application Programming Interface, these are a set of functions for use by developers.

**IDE:** Integrated Development Environment, usually consists of a source code editor, build automation tools, and a debugger.

**Functional Requirements:** These requirements describe what the application and website do within the system. They should be clearly defined with inputs and outputs.

**GeoJSON:** a format for encoding a variety of geographic data structures.

**Geotiff:** is a standard .tif or image file format that includes additional spatial (georeferencing) information embedded in the .tif file as tags.

**NAU:** Northern Arizona University.

**Non-Functional Requirements:** These requirements describe how the application and website work. They should be clearly defined with inputs and outputs.

**Raster data:** is used in a GIS application when we want to display information that is continuous across an area and cannot easily be divided into vector features.

**SDK:** Software Development Kit, is a group of software tools used for development.

## 10. Appendix A: Development Environment and Toolchain

10.1 Software

**Map**

The map was developed and tested using the Mapbox that runs on browsers. Required hardware for the application development is a PC or laptop with Windows installed. Windows 10 was used during the development.

**Website**

The website was developed mostly on Visual Studio and Dreamweaver using Google Chrome to test the website locally and see immediate changes. Required hardware for the website development are:

• A PC or laptop with Windows installed. Windows 10 was used during development.
• A text editor to edit HTML5, CSS, and JavaScript files. Visual Studio and Dreamweaver were used during development.
• An internet browser to help test and develop the website. Google Chrome was used during development.

10.2 Toolchains

MapMapbox studio: an online map design studio developed by Mapbox.

Mapbox GL JS:  a Javascript library to render an interactive map from Mapbox styles.

Google Chrome: a web browser to access Mapbox.

HTML: Hypertext Markup Language, is the standard language used for development on the World Wide Web.

CSS: Cascading Style Sheets, is used to format HTML5 elements.

JavaScript: This is an object-oriented programming language used to make web pages more interactive.

Gdal:  a computer software library for reading and writing raster and vector geospatial data formats. This is used to color the map with a range of values.

Python: an object oriented programming language that gdal uses

Visual Studio: This is a text editor for editing source code files of HTML5, CSS, and JavaScript.

10.3 Setup

Our work is mainly based on Mapbox Studio (https://account.mapbox.com/). Mapbox Studio is supported in browsers that support WebGL, a method of generating dynamic 3D graphics using JavaScript. Mapbox Studio is compatible with all modern browsers.

- To access mapbox studio you can login with our provided account id and password.
- Once login, click studio under the account logo.
- You can see a list of our tilesets from 2010 -2015.
- To start modifying our value just click on any tileset you desire.
- To add a layer click over to the Layers tab. Click + in the upper left. In the list of Data sources, click the name of the tileset, and then click the source layer that appears to add it as the source for the layer.

To use our code, install any editor. In this project we use Visual Studio, you can download here https://visualstudio.microsoft.com/downloads/.

10.4 Production Cycle

Upload a new tiff image to Mapbox Studio

- Go to Tilesets -> New Tileset -> choose your tiff image ( only one at a time)
- You cannot rename your tileset once it is uploaded. What you can do is name your tiff image to avoid confusion later.

Create a style. A style is a JSON document that defines the visual appearance of a map. It can hold many layers (tilesets)

- Click on new styles-> choose a template -> customize template. If you don't like any temple you can choose a blank template to start from scratch.
- Once a style is modified, go to the share panel. There are a few options here for you to choose

    - Make the styles public or private
    - Download style as zip file including Stylesheet (JSON), icons (SVG), and fonts (TTF or OTF)
    - Copy style URL or access token
- Click publish when everything is done to allow other people see the map and its changes.

To edit a HTML, CSS, or JavaScript source code file, you should navigate to the directory containing the file and open the desired file in your preferred text editor.