Design Documentation - Group 15 - InfoVis
Group Members: Berkley Hughes, Emma Broggi

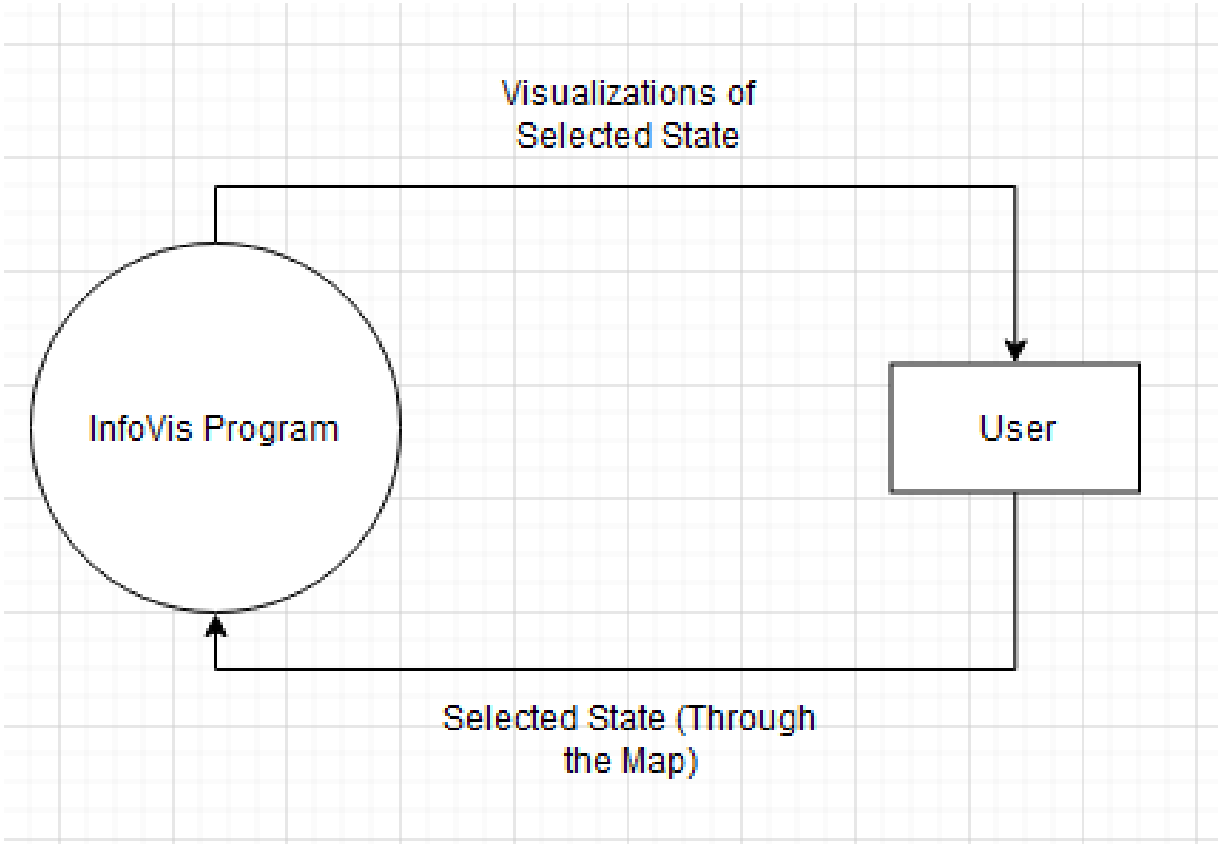
Project Overview:

This project attempts to answer the question, “what is the best state for me to live in?” This is done by providing the customer with different metrics for each different state, allowing them to compare and contrast states against one another. The metrics are provided using different data visualizations created using the D3JS library and data from multiple different csv files. There is only one main actor in the system as the program is embedded in a webpage and there is not a required connection to an external database.

User Stories and Product Backlog (Linked to Actual Spreadsheet):

# Number	Front	Back	Priority
0	As a <user who wants this fucntionality> I need a <what the user wants> So that <why the user wants it>	Acceptance Criteria <what the user should be able to do with the functionality>	#
1	As a User, I would like to be able to see Immigration and Emmigration rates, So I can know where people are moving to and from.	Acceptance Criteria: User should be able to see a chart of immigration/emmigration rates, and where the people are coming from and going to.	4
2	As a User, I would like to be able to see Crime rates of each state, So I can know how safe each state is.	Acceptance Criteria: User should be able to see a graph of both property and violent crime rates for the selected state.	3 (tied)
3	As a User, I would like to be able to see the employment ratesof each state, to know where possible job opportunities could be.	Acceptance Criteria: User should be able to see a graph of both employment rate for the selected state.	3 (tied)
4	As a User, I would like to be able to visually know which state I am selecting, So I can easily refer back in case I have forgotten.	Acceptance Criteria: User should see a map of the United States that highlights the selected state when clicked on.	1
5	As a User, I would like to be able to know the cost of living in each state, So I can know if I would be able to afford to live there	Acceptance Criteria: User should see a graph showing the cost of living of the selected state.	2
6	As a User, I would like to be able to see the trends of the data metrics, So I can know which places are getting better and which are getting worse.	Acceptance Criteria: Every graph should use data from the previous decade to show the overall trends of the metrics.	5
7	As a User, I would like to be able to be able to see the Temperature/climate of each state, So I can know if I would enjoy living there.	Acceptance Criteria: User should see a circular graph showing climate trends throughout the year.	6
8	As a User, I would like to be able to see tax rates for each state, So I can know what it costs to live there.	Acceptance Criteria: User should see a chart showing the tax rates for the selected state.	7

Simple Context Diagram:

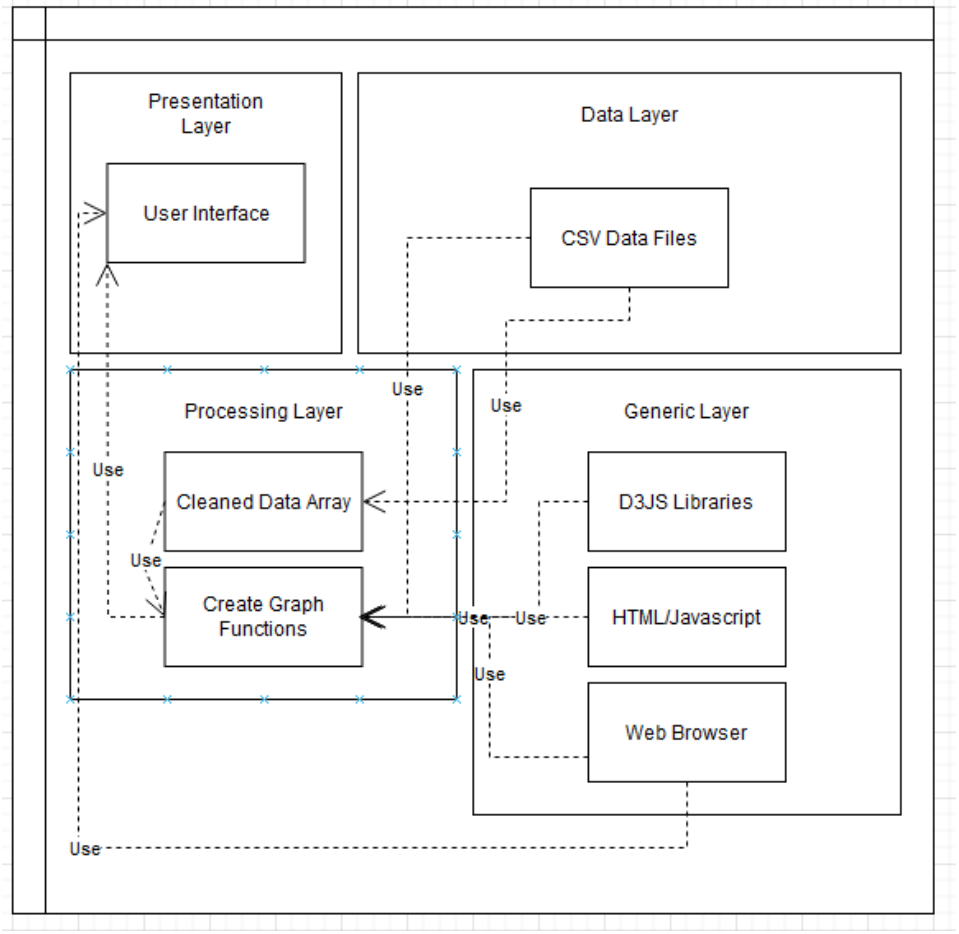


Architectural Overview:

Introduction:

There were a couple alternatives to the design we chose to go with for the final product. The one that got the furthest along the line was using an API to read data from webpages to get the desired data for the states. The design would fall under the Monolithic style, as everything is contained within one application and doesn't need an external server to run. This design has some merit to it, as it would bring the most up to date information to the customer. However, we were not accustomed to using the APIs; it would make finding trends in the data much harder as the data would need to be cleaned and then stored back into a new database along with other csv data files that provide the older data points. We thought just using CSV files would make the process much smoother and we would end up with a better, more fleshed out product by keeping it simple.

- Subsystem Architecture
    - UML Package Diagram:



The subsystem architecture for the program is rather simple, as there is no required internet connection and the data is being pulled from CSV files in the application folder. This means the design is very easily broken down into 4 different layers shown in the diagram. The Presentation layer contains what the user will see when using the program. The Processing layer is what is done to clean the data and create the graphs that will be used by the user interface. The Data layer is just the CSV files used for the data in question and the Generic layer is all the libraries and coding generics required for the program to function.

- Deployment Architecture:

  This software will run on a single processor.

- Data Model:

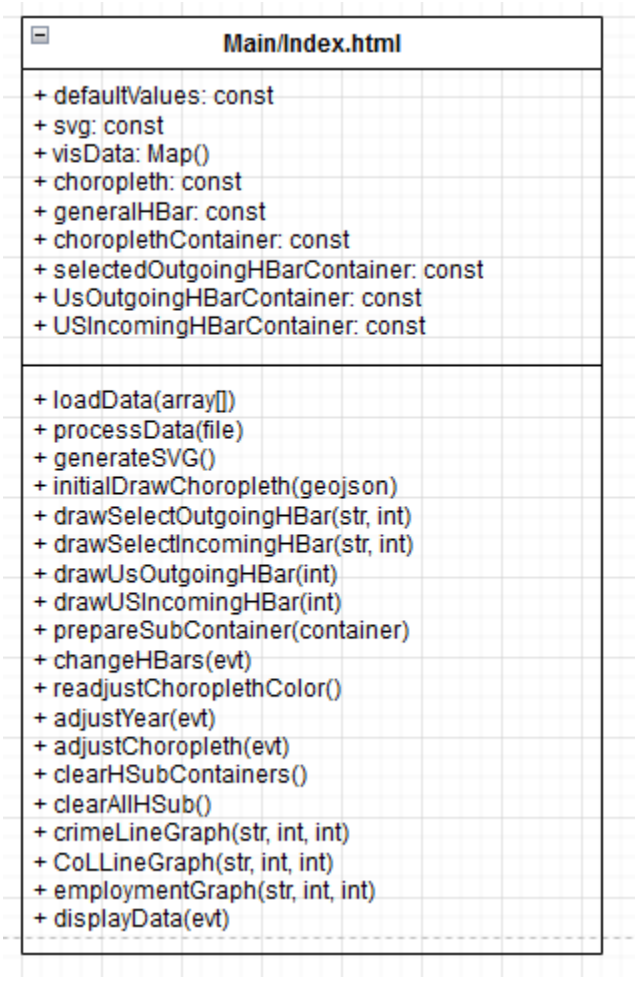
The persistent data storage for the program is multiple CSV files. The data in the files is never changed, just pulled from the files. The files used include migration data, state taxes, unemployment rates, average monthly temperatures, county map data, US state map data, crime rate data, and postal code data. Each file has its own way of storing data that is cleaned and added to the data map when the program is running. After the program exits, the data map is discarded.

- Global Control Flow:

All functions besides the initial boot up of the program are event driven. All the visualizations are shown based on which state the customer has clicked on. This is highlighted in orange to visually show which state is being represented. The system initially starts by showing the data for Alaska and changes whenever another state is clicked on the map. The user can also change the year shown on the map by clicking on the specific year, or which metric should be highlighted on the map. There are no time dependencies in the system. The system runs on a single thread of control.

Detailed System Design:

- Static View:

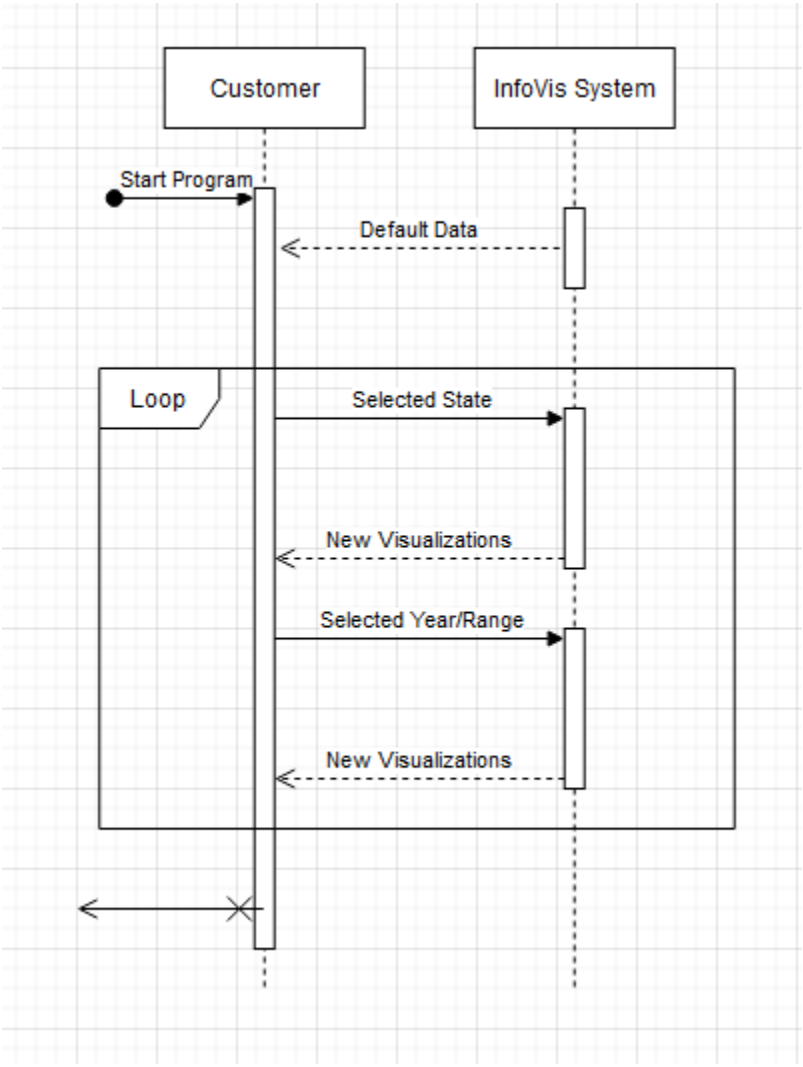


Because the code of the program is contained within one file, the UML class diagram becomes quite large and a bit confusing to look at. For the variables, all the values are type const besides visData which is a map. This is because they are collections of important data points that get accessed later in different functions so they act like a file system. An example would be defaultValues. It contains variables chosenYear (int), chosenState (str), startYear (int), endYear (int), and chosenStat (str). Many of the other variables include D3JS controls and parameters for drawing the visualizations.

Many of the functions do exactly what their name entails. None of the functions return any values as they either change global variables, such as visData, or draw something onto the screen. For the input data types, the array in loadData is a url. This is because D3JS requires a server connection to run, but that server is just the local host. The file in processData is whatever csv file is being accessed. GeoJson is a special type of json file that contains geolocation data for drawing out maps; in this case it is a map of the United states. Container is the area created for the visualization by javascript that gets included in the HTML file. Evt is an event handler variable. This can be something like a mouse hover, but in this instance, all the evt variables represent the value given on a mouse click.

- Dynamic View:

  UML Sequence Diagram:



Future Test Plans:

Currently, we do not have visualizations for all of the metrics we have data for. The two missing are temperature and taxes. These have been a bit difficult to make work as the ideal visualizations for temperature is very difficult to create and taxes can be all over the place. Currently, the difficulty along with time constraints has been the main issues to figuring these out and there is no guarantee that we will be able to complete them before the due date.