NAU NORTHERN ARIZONA UNIVERSITY
School of Informatics, Computing, and Cyber Systems

Software Design
Version 1

Date: September 20th, 2024
Team Name: FairyMander
Sponsor: Bridget Bero
Professor: Isaac Shaffer
Mentor: Vahid Nikoonejad Fard

Team Members:
Izaac Molina (Team Lead)
Dylan Franco
Jeysen Angous
Sophia Ingram
Ceanna Jarrett

Table of Contents

1 Introduction

Congressional district lines are redrawn following the release of the census, a survey that is taken every ten years. The census provides information on population, demographics, economics, geography, etc. The number of seats each state gets in the US House of Representatives is determined by the state's population, thus determining the state's number of congressional districts. Congressional districts are the voting districts for state representatives, which are members of Congress. These districts must comply with state and federal laws, and represent the population. Every two years, people within each district vote for their representative in Congress. This is an important decision since Congress holds all legislative power; it is the only part of the government that can make and change laws.

Currently, thirty-nine state's redistricting processes are controlled by their state's legislative body. In these states, district lines are manipulated and approved by their members of Congress. The first draft of the proposed district lines are drawn by a legislative committee. Members of the legislative committee are chosen by the legislature. Some of their responsibilities include: monitoring government operations, recommending courses of action to the Senate, etc. The draft of the proposed districts may be vetoed by the governor, but this veto can be overridden by a ⅔ legislative vote. Final approval of the proposed districts varies by state, but it is most commonly approved by legislature with a ⅔ vote. However, the legislature holding all the power to create new districts causes a problem to arise.

Gerrymandering is the act of drawing district lines in a way that benefits certain political parties; this leads to underrepresentation and political manipulation. Representatives have the power to manipulate electoral outcomes during redistricting, thus guaranteeing their electoral success. This power poses a threat to our democracy, having the potential to make voting obsolete. Gerrymandering is done through packing and cracking; techniques used to dilute votes. There are active solutions in place for gerrymandering, but they are not as widespread as they should be. One solution against gerrymandering is for voting districts to be drawn by the Independent Redistricting Commission (IRC). The IRC is separate from the legislature, meaning that politicians do not have the opportunity to manipulate voting districts. This makes the redistricting process fair and transparent. The IRC must adhere to certain criteria when creating districts, such as equal population, protecting racial minorities, and making districts compact. However, only eight state's districts are currently being drawn using this system. This means the other 42 states' voters are being gerrymandered and suppressed.

In search of a more widespread solution, our client, Dr. Bero, reached out to us to create an open-source, non-biased redistricting algorithm. Our team goal is to create a user-friendly website to educate citizens on how congressional districts can be created fairly by utilizing our algorithm. The results of running our algorithm on each state will be added to our website along

with a description of what makes them fair. Through this, we hope to educate the public and leave an impact on our current voting system.

2 Implementation Overview

FairyMander will be composed of two primary components. The first is a Python package for generating “fairymandered” district maps, and the second is a website with an interactive US map to display these maps, along with supplemental information about redistricting laws and the metrics used to calculate district fairness.

2.1 FairyMander Python Package

An essential goal of this project is to generate state district maps and evaluate their fairness. To do so, we will first need to acquire the data needed for redistricting calculations, such as state geography, population, and voting outcomes. This data will be aggregated from the official US Census as well as the Redistricting Data Hub (RDH), which is a well-respected organization dedicated to providing data for redistricting. The Census will provide data on geography, populations, and racial distributions, and the RDH will provide data on voting outcomes.

With this data collected, we will be using Python to develop these redistricting utilities. Along with its relative ease of use, Python features numerous useful libraries for processing and outputting geographic data. GeoPandas, a library for processing geographic data, contains several useful utilities for our needs, such as processing shapefiles, which are a geographic data file type used by both the Census and RDH. Additionally, GeoPandas can output this data into a visual map. However, we also want our maps to have interactivity so our users can examine data related to the generated districts. To accomplish this, we will be using Folium, a package that can make GeoPandas maps interactable. These interactive maps can also be embedded into a web page, allowing us to display the results of our algorithm on the FairyMander website.

Beyond this, pre-existing packages exist to cluster geographic units based on population. For our purposes, we will be using PySal, a common library for analyzing geospatial data. It provides numerous useful utilities for geospatial analysis, such as assigning weights to each geographic unit and prepackaged clustering algorithms that can be used to cluster geographic bodies. Such tools will be massive time savers both in providing already established utility and in using pre existing algorithms to build our work on.

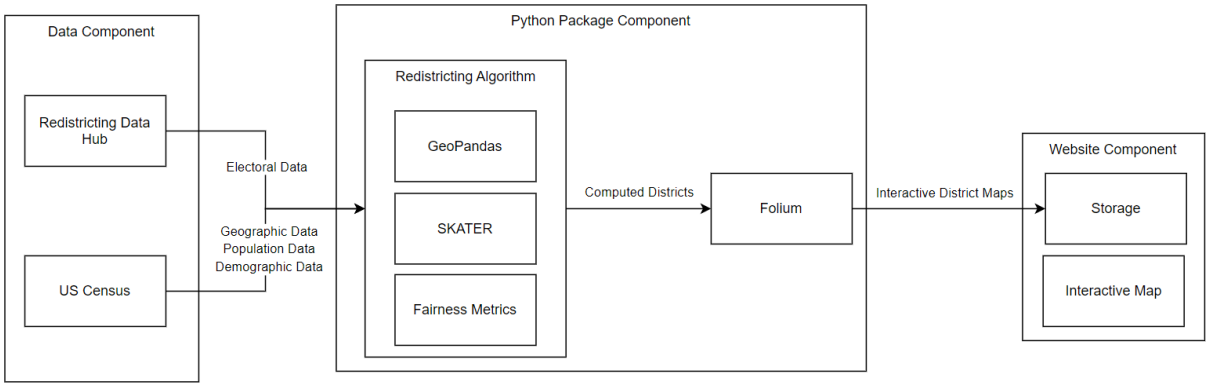
Finally, we will have a set of fairness metrics that we will use to evaluate the districts for fairness, which will be defined in a Python module. These metrics will be derived from previous redistricting research and district evaluation protocols.

2.2 FairyMander Website

The maps generated using our Python package will be served via an interactive map of the U.S. on a public website. This website will be served using common web development tools, namely HTML, CSS, and Javascript. When users click on a state, they will be redirected to that state's "fairymandered" map, as well as its current district map. On this page, users can interact with the Folium district maps (see 2.1 more info on Folium), inform themselves about the redistricting metrics we used to evaluate these maps, and learn more about the redistricting policies for the selected state.

3 Architectural Overview

As described in Section 2, our software will take in data from the US Census and the Redistricting Data Hub, use this data to execute our redistricting algorithm and then serve our results on a website. The general Architecture for this process is outlined below:



3.1 Data Component Architecture

The Data component of the project consists of downloaded files from the U.S. Census and Redistricting Data Hub (RDH). For this project's needs, a one-time download for this data will suffice, as there is no need to pull real-time data using the Census and/or RDH APIs. This data will then be "cleaned" by removing unnecessary table entries and unifying the Census and RDH data into an aggregated data file for each state. While it's possible this approach could be automated by developing a separate package, the data component's main purpose is to provide an example dataset to test our redistricting algorithm, which is the primary focus of the project. Therefore, while such a package would be useful, it is out of scope for this project.

3.2 Python Package Component Architecture

From here, the Python Package component will take in the data to compute and evaluate state district maps. The data obtained from the Data component will be read in using GeoPandas, which SKATER can then utilize to generate district maps, which will also be stored using GeoPandas dataframes. As the maps are generated, they will be evaluated using a set of utilities in the Fairness Metrics module, which will use the generated GeoPandas dataframe to perform a series of calculations for evaluating district fairness. These metrics will then be aggregated into an overall composite "fairness" score. This process will repeat until a user specified number of districts to create is reached, at which point the top n districts (where n is also specified by the user) will be kept based on composite fairness score. From here, these kept districts will be converted into interactive maps using Folium, producing a set of final maps and saving them to the local file system.

3.3 Website Component Architecture

With the maps generated, our team will then examine the maps and analyze their various pros and cons using best redistricting practices. This process will reflect the real life process that redistricting officials would go through when using computer generated districts in order to consider the nuances between each state's redistricting needs. Once a map is chosen, we will then upload the map into the FairyMander website, where it will be integrated into the interactive map. From there, any accessory information about the state's redistricting laws and our justification for the redistricting of said district will also be input manually by the team. This website then acts as the primary way in which users will interact with FairyMander, showcasing the project's results and educating users on redistricting practices.

3.4 Architectural Style

Due to the unique needs of our project, we needed to adopt an architecture that would allow a clear and defined data flow for obtaining redistricting data, creating districts from that data, then presenting the created districts. As such, our software most closely aligns with the 'Pipe-Filter' architecture style. This architectural style focuses on transforming data by sending it through defined 'pipes' and processing it at defined 'filters', leading to a desired end result. In our project, each component is a 'filter', which will take in data through 'pipes' as the redistricting data is transformed into interactable district maps. This approach allows us to keep each 'filter' highly modular, decreasing coupling between the different components and subcomponents of the project. It also allows flexibility as at any point in the pipeline, a new 'filter' could be added to further enhance the generated districts. However, it is worth noting that a key next step for FairyMander would be to further develop the 'pipe' parts of our project. The current scope does not define modules for automating the transfer of data between certain components, such as the consolidation of data between the RDH and Census, as well as the upload of maps directly from

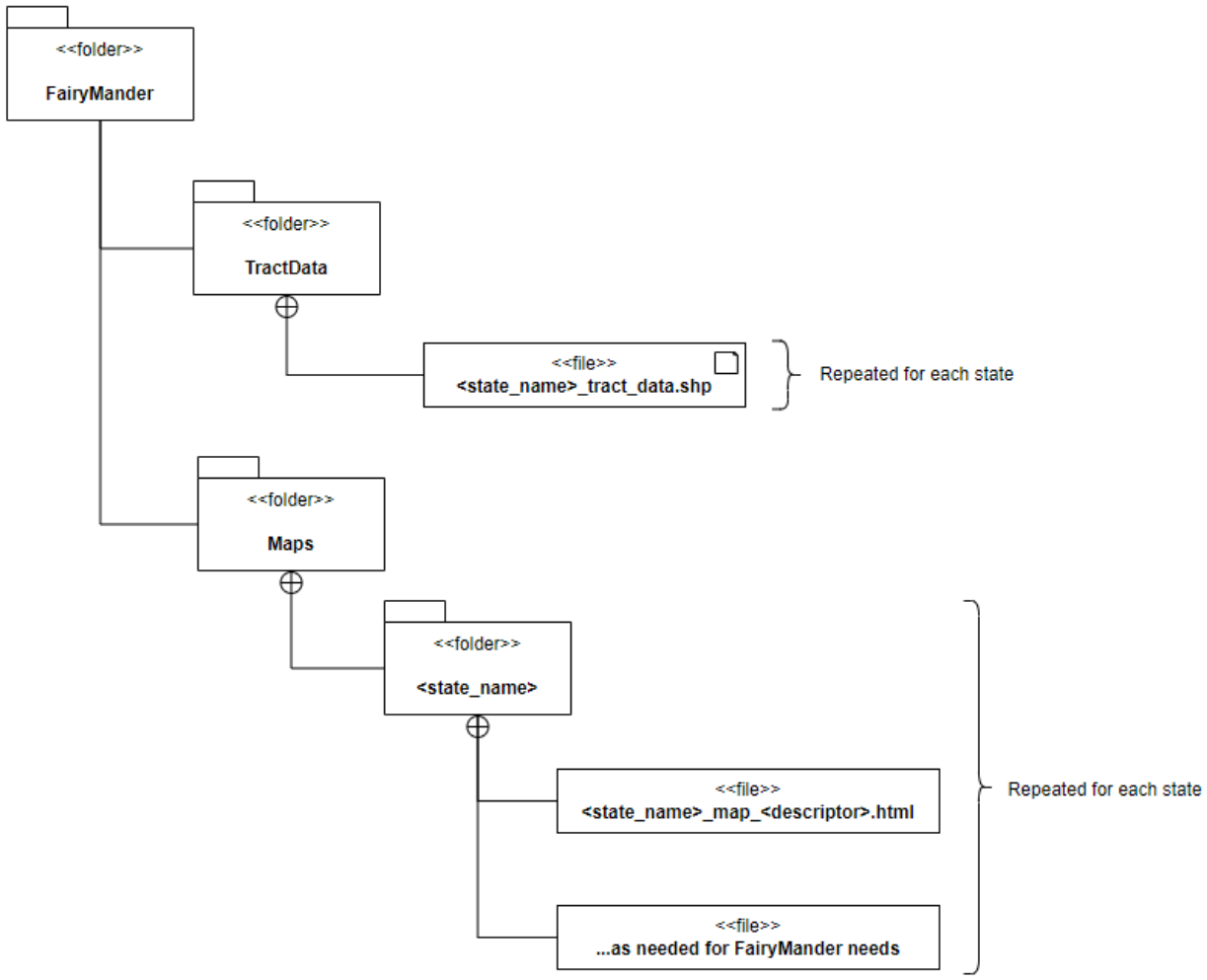
Folium to our website. Such ‘pipes’ could greatly enhance the efficiency at which FairyMander could operate, and would be an appropriate next step upon completion of this project.

4 Module and Interface Descriptions

4.1 Data Component Description

4.1.1 File Structure

The Data Component is primarily a logical distinction, as it is simply a collection of files to be used as sample data for the python package. The data component follows the following structure:



In this structure, the root folder will have two directories: TractData and Maps. TractData is used to store the input for our algorithm, as it will hold census tract data for each state in a .shp or

“shape” file, a common data format that the Redistricting Data Hub, Census, and Geopandas all use. In the Maps directory, Folium output maps will be stored as needed for the project's needs, as we may want to store multiple maps for the same state to compare results in the future. Each state will have its own folder for these generated maps.

4.1.2 Shapefile Datafields

When looking at the data, different data fields will need to be gathered. These data fields will be used to calculate the fairness metrics we will use to evaluate districts. Through ongoing research, we have acquired and will continue to acquire these metrics, resulting in a list of fields that will be used to perform these calculations. The shapefile data fields that will be collected will be the geographic data, Democrat/Republican registered voter, total registered population of each party, the total population of the district, and the population for each racial group in the district.

4.1.3 Compiling the Data

We use three types of data to gather the information we need. First is voter registration data, which is available for each state and organized by census block. Second is census tract data, which lists the census tracts that make up each county in every state. Lastly, we use census block data, which connects the first two by showing which census blocks belong to each tract within a county and state. To prepare the data, we process it in two stages. In the first stage, we combine the census block data with the tract data. This helps us organize the voter registration data by location. In the second stage, we merge this combined data with the voter registration data, resulting in a file for each county in each state, sorted by census tract and containing all relevant voter data.

To achieve this, we wrote several programs in C and Python. The first program was a Python web scraper that efficiently gathered the necessary census tract and block data from www2.census.gov using a recursive algorithm to download only the required files. Next, a C program combined these two datasets (census tract and block data) for each county and state. Finally, another C program merged this output with voter registration data gathered manually from redistrictingdatahub.org. The resulting files are now ready for integration with redistricting shapefiles.

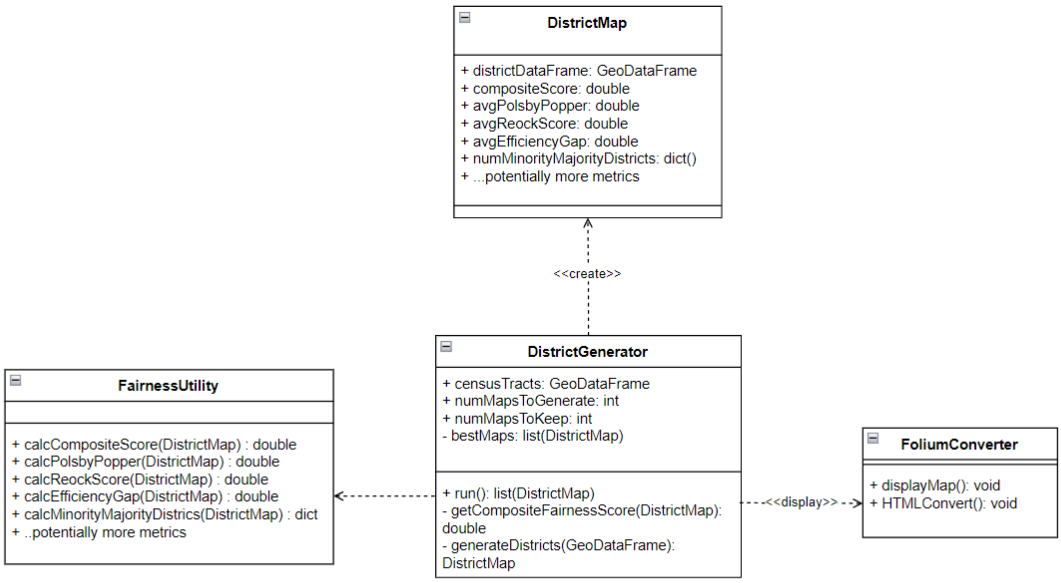
4.2 Python Package Component Description

4.2.1 Definitions

In this section, we will be using a number of terms and metrics related to redistricting. For reference, here is a table containing definitions for these terms:

Name	Definition
Polsby-Popper Score	Metric for district compactness. The ratio of the district's area to the area of a circle whose circumference is equal to the district perimeter.
Reock Score	Metric for district compactness. The ratio of a district's area to the area of a minimum bounding circle that the district can fit in.
Efficiency Gap	Metric for district partisanship. Determined by calculating the ratio of the difference between the dem/rep votes and the total votes.
Minority Majority District	A district where a particular minority group has %50 or more representation.

4.2.2 Class Diagram



The FairyMander Python package consists of 3 modules: the primary 'DistrictGenerator' Module for running the redistricting algorithm, the 'FairnessUtility' module for performing fairness metric calculations, and the the 'FoliumConverter' module for converting the generated district maps into interactive Folium maps. Additionally, the 'DistrictMap' dataclass is used to store information for our generated maps. In the following sections, we will describe each class and the public interface it provides to other classes in the package

4.2.3 DistrictGenerator

DistrictGenerator acts as the main access point for generating districts. When a developer wants to generate a set of districts, they will create a DistrictGenerator object, initializing it with a geopandas dataframe (GeoDataFrame), a number of district maps to generate, and a number of maps to keep from the generated ones. They then call 'run()' to execute the generator, returning a list of district maps.

An example of this process is shown below:

```
import DistrictGenerator from generator
import geopandas

# load data from AZ shape file
az_shape_data = geopandas.read_file("arizona_census_tracts.shp")


# init generator, in AZ generate 100 maps, keep the top 5
my_generator = DistrictGenerator(az_shape_data, 100, 5)

# generate districts
districts = my_generator.run()
```

While this is the extent of the module's public interface, it will also internally utilize the FairnessUtility module to evaluate each district map for the described fairness metrics. It will also contain a private method that utilizes PySal to generate a district map.

4.2.4 DistrictMap

The DistrictGenerator will generate DistrictMaps, which are a specialized dataclass used to store needed information for each district map. The map is defined by a GeoDataFrame, which contains data for each generated district, such as its geometry, population, and fairness metrics for each individual district. The remaining attributes are aggregated fairness metrics, to see more on these metrics, refer to 4.2.1 and 4.2.5. This data class not only creates a simple way to easily access these metrics, but also is extensible to more fields so the FairyMander team can add additional fairness metrics if needed.

4.2.5 FairnessUtility

To effectively evaluate generated district maps, we must build out utilities that can compute district fairness metrics given a GeoPandas dataframe. Such utilities are to be utilized by DistrictGenerator objects during the redistricting algorithm runtime. This class of functions will be used to compute individual scores for each redistricting metric we will use, as well as a

function that computes an overall composite score based on the metrics calculated for the given District Map. It is critical that this module is kept as extensible as possible, as we want the project to be consistently open for extension to other redistricting metrics.

4.2.6 FoliumConverter

The FoliumConverter will use Folium for visuals on the web application. It will use GeoPandas data storage to create a folium map object with its location centered using latitude and longitude. Moreover, Folium will be used to display details of a certain district when hovering over it and can add extra details about a map as needed. Lastly, the map will be converted to an interactive HTML file and displaying the map using a <iframe> tag for user interaction and satisfaction.

4.3 Website Component Description

4.3.1 Responsibilities

The website must include numerous responsibilities and features that will maximize user-friendliness and efficiency which will consist of aspects such as:

4.3.1.1 Landing Page

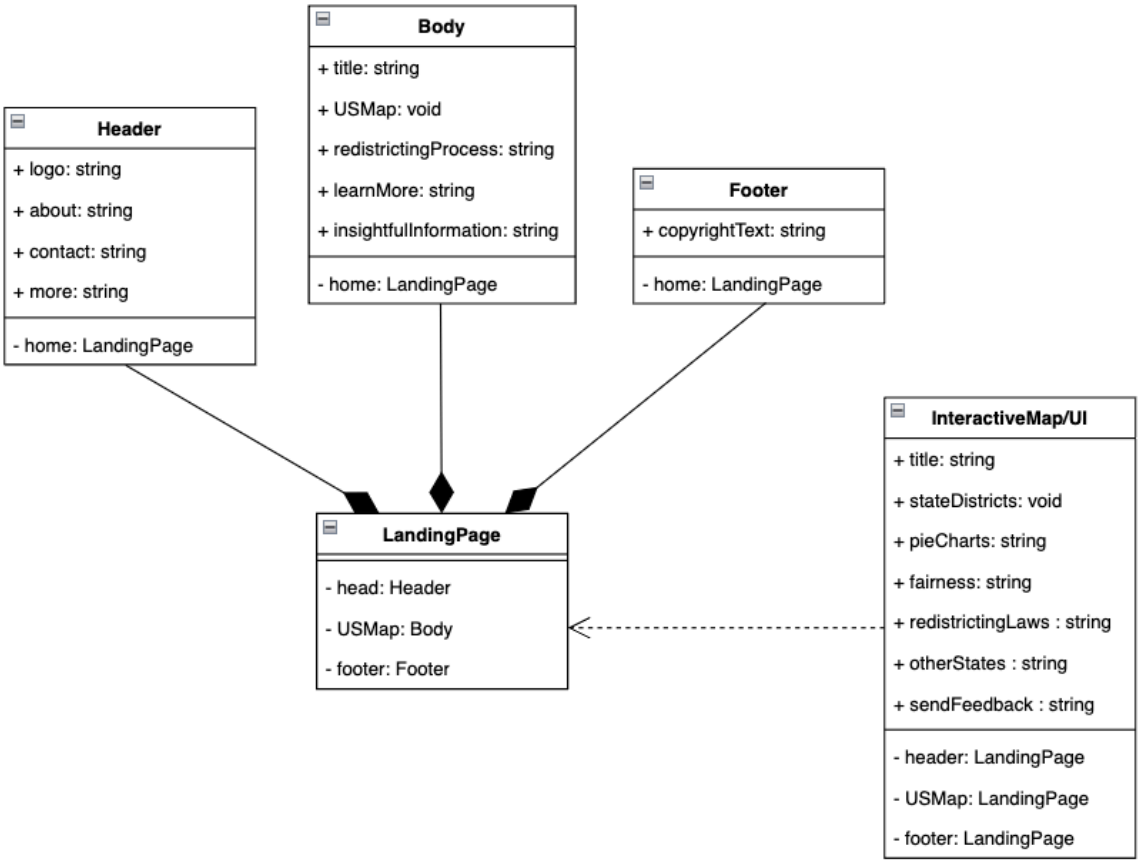
The landing page will consist of the header, body, and footer. The header will include navigation tabs like home, about, contact, and more. The body will include an interactive United States map where users can hover over a certain state and will be able to see the state's name, population, and how many seats/districts are in that state. Moreover, under the interactive map, users can read about the redistricting process, learn more about our mission, and get some insightful information that can be helpful. Lastly, the footer will contain copyright notice.

4.3.1.2 Interactive Map/UI

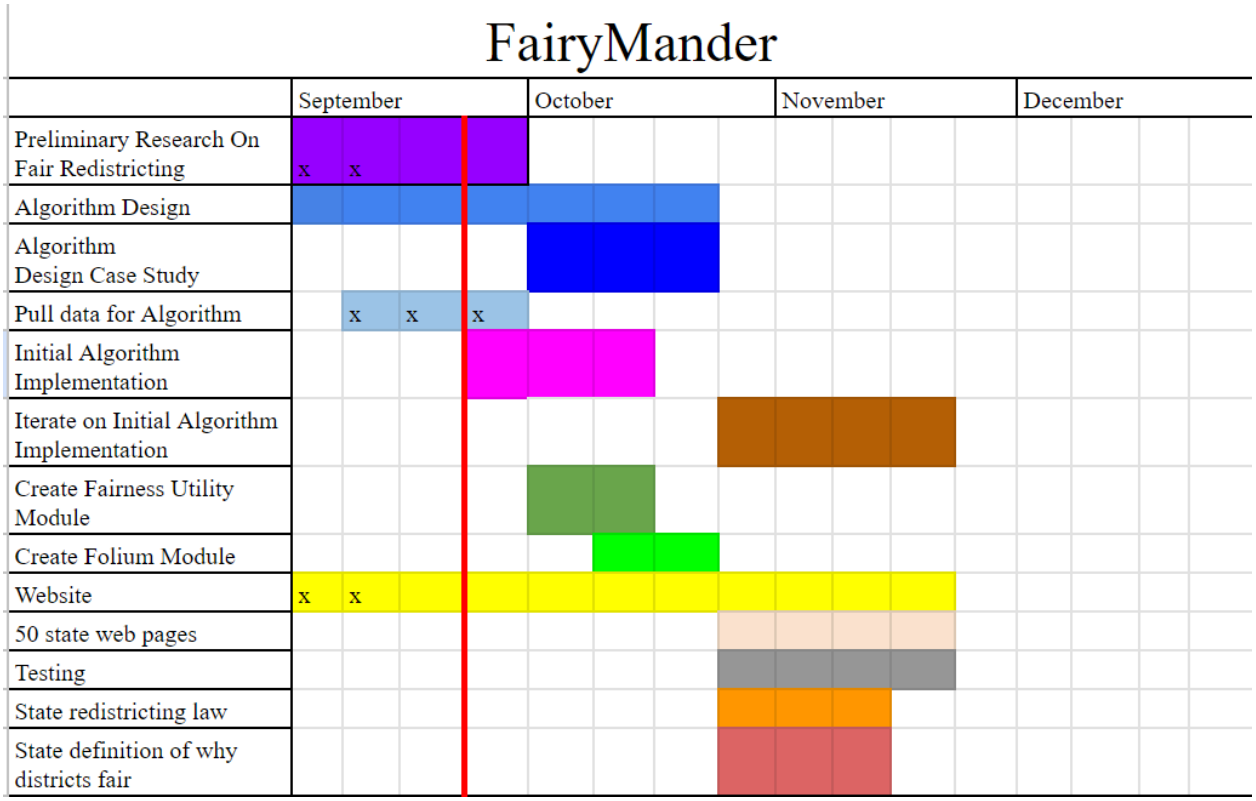
Each state will have its map which uses folium to make the map interactive and user friendly on the web application. The map will display each district in different colors, once hovered, the district number will be displayed along with the population, Democratic and Republican vote percentage in that district. Additionally, a dropdown menu will feature pie charts which represent the race distribution for each district. Furthermore, users can read about understanding why our district is far, redistricting laws that our team followed while creating the algorithm for each state, and provide feedback if they wish too. Lastly, users can either go back to view the United States interactive map or choose another state as all fifty will be listed in alphabetical order.

4.3.2 Structure

The website component follows the following UML class diagram:


Body
+ title: string
+ USMap: void
+ redistrictingProcess: string
+ learnMore: string
+ insightfulInformation: string
- home: LandingPage
Header
+ logo: string
+ about: string
+ contact: string
+ more: string
- home: LandingPage
Footer
+ copyrightText: string
- home: LandingPage
InteractiveMap/UI
+ title: string
+ stateDistricts: void
+ pieCharts: string
+ fairness: string
+ redistrictingLaws : string
+ otherStates : string
+ sendFeedback : string
- header: LandingPage
- USMap: LandingPage
- footer: LandingPage
LandingPage
- head: Header
- USMap: Body
- footer: Footer

5 Implementation Plan



The implementation plan for FairyMander is laid out in the Gantt chart above, with the red line being the current timeframe. At this point, the team has nearly completed the initial data cleaning process, as the majority of our shape files have been assembled. Following this, the team has started developing the initial implementation of our algorithm using strategies in pre-existing research papers. We aim to refine these strategies and apply them to a single state case study on Arizona, which we will use to guide our initial algorithm implementation. In tandem with this, we will develop the Fairness Utility and Folium Modules using mocked district Map Data. Additionally, refinements will be made to the website as the team gets more familiar with what our results will look like. This development process will complete our “Alpha” version

After this, we will enter the refinement, map completion, and testing stage. In this stage, we will continue to make refinements in our district generation methods as well as the fairness metrics used. In tandem with this, we will use our algorithm to generate districts for the remaining states so we may display our results. As these maps are generated, we will gradually display them on our website along with descriptions for why the generated maps are deemed fair. As this functionality is built out, we will also implement unit and integration tests using Python’s built-in unittest module. Collectively, these processes will continue up to the end of our project timeline.

To conclude this section, the following table elaborates on the roles we currently see each member playing in the project:

Name	**Role/Responsibility**
Izaac	Keeping track of the team's progress and leading meetings, setting up/managing python packages, developing district generator module.
Dylan	Data cleaning, developing the fairness module, flex developer for other modules.
Ceanna	Researching fairness metrics, developing fairness module, keeping track of important team information, detailing state laws for website content.
Jeysen	Repository manager, website manager responsible for setting up the interactive map and resulting district pages, developing folium module.
Sophia	Primary developer for algorithm implementation using PySal, developing district generator module.

Additionally, due to the sheer number of states, each team member will likely be responsible for performing tests using our algorithm on each state for our final map

6 Conclusion

To conclude, FairyMander is dedicated to creating effective computer generated solutions for redistricting plans. The design outlined in this document provides a scaffolding for the development of this project as we continue to develop and research a fair and balanced redistricting algorithm. We first acquire and clean data from the U.S. Census and Redistricting Data Hub to obtain up to date, high quality population, voting, and geographic data for all Census tracts, forming the data component of our project. Then, this data will be fed through a redistricting algorithm as part of a developed python package, generating district maps and evaluating said maps for their fairness. Finally, we will convert these results into interactive maps that will be embedded into our website. By highlighting a clear data path, promoting high levels of extensibility, and translating our results into clear visualizations, we will effectively

solve this problem in a way that is both easy for our users to understand and for our developers to maintain. While the path ahead has many challenges, our team is ready to tackle these challenges head on and will use this document as a foundation for the implementation of this project.

9 References

[1] "State-by-state redistricting procedures," *Ballotpedia (n.d.)*, https://ballotpedia.org/State-by-state_redistricting_procedures.

[2] U. C. Bureau, "Population," *Census.gov*, 22-Jul-2022, https://www.census.gov/topics/population.html.

[3] B. LITTLE, "How gerrymandering began in the US," *History.com*, 20-Apr-2021, https://www.history.com/news/gerrymandering-origins-voting.

[4] "The House Explained," *United States House of Representatives. (n.d.),* https://www.house.gov/the-house-explained.

[5] "The Legislative Branch," *The White House*, 15-Jan-2021, https://www.whitehouse.gov/about-the-white-house/our-government/the-legislative-branch/.

[6] "Who draws the lines?," *All About Redistricting*, 04-Aug-2021, https://redistricting.lls.edu/redistricting-101/who-draws-the-lines/.

[7] "About the committee system: Committee assignments," *U.S. Senate: About the Committee System | Committee Assignments*, 08-Sep-2023, https://www.senate.gov/about/origins-foundations/committee-system/committee-assignments.htm

[8] S. J. Eckman, "Congressional redistricting criteria and considerations," *Congressional Research Service*, 2021, https://crsreports.congress.gov/product/pdf/IN/IN11618.

[9] "Redistricting," *Ballotpedia (n.d.)*, https://ballotpedia.org/Redistricting#Background.