DataWin: Elaboration Iteration II Report

Open Data Access Portal by Nidhi Patel & Niraj Patel

University of Windsor

Dr. Kobti

November 30, 2020

Shortcut Link:

Code Repository: https://github.com/nidhipatell/WinData

The above link will take you to the main code repository where you can access everything else.

Bug Reporting: https://github.com/nidhipatell/WinData/issues

The above link will take you to the issues form where you can view all the tickets created for the issues the team encountered.

Project Management: https://github.com/nidhipatell/WinData/projects/1

The above link will take you to the GitHub project management for WinData.

Documentation Repository: https://github.com/nidhipatell/WinData/wiki

The above link will take you to the documentation repository where you will find all the wiki.

Testing Strategy: https://github.com/nidhipatell/WinData/tree/main/Iteration%20II/testing

The above link will take you to the testing strategy used by the team. Please read ahead, the report talks about why and how we chose our testing strategy.

Acknowledgement

The completion of the project WinData would have not been possible without a few people such as our professor Dr. Ziad Kobti and our friends who have really helped us shape our application today the way it is. With their input we were able to create a better user interface and user experience. Few online resources have been great help and they will be listed in the reference page.

WinData is a joint effort from Nidhi Patel and Niraj Pate. Nidhi Patel was mainly in charge doing the frontend. She was responsible for files such as HTML, CSS, and Bootstrap. While Niraj was mainly in charge of the backend and data visualization.

Abstract

The problem our team aims to address with the WinData, a web application, is to

provide an additional feature to users where they can see key data relevant to a specific dataset without having to download it. This will ensure that it saves the users time and overall increase their efficiency and will help them decide if a particular dataset is right for them or not. Our website has multiple dataset which has .csv files or .xml files and they can select a dataset of their choice and view the graphs. As additional key feature they are also able to download the files if they would like to. As of now there is no competitor for this product, this application is a mixture of data visualization and data portals. It is different from other data portals since they do not provide this key feature that our application provides that is, data visualization. Furthermore, we aim to create a fully functional website which contains multiple dataset that users can browse through.

Keywords

1. MVC model:

   For our project we decided to follow a specific architectural pattern and that pattern is Model-View-Controller (MVC). The MVC model is mainly used for developing modern user interfaces. The Model represents all the data-related logic that the user will be working with. It is the data that is transferred between the controller and view components. View component is used for all the UI logic, it is what represents the data, it gets the data from the model and displays it to the user. The last component is controller, this component controls the data of the mode, it would update the model when requested. In our case model is all the .csv and .xls files that represent the data. View is represented by HTML, CSS files which are both separated into **static** and **templates** folder. Controller is the python file which is the backend of the application, this file is called **__init__.py** this routes the website and is in charge of displaying the data when requested.

2. Request/Response Cycle:

   The website circulates between a request-response cycle. A request is made by a user. A user who access the web content which is displayed by the server and the HTML/CSS files. When the user navigates the website, it is always routed by the backend file called __init__.py which will route the website and ultimately display all the web content. When a user selects a data base and then clicks on the button to visualize the dataset. The backend will use the .csv file and using **matplotlib** it will graph the data. This will then be routed to the web content where it will be displayed to the user right away. For this the website will make a “POST” HTTP request that will initiate all this process.

3. Event-driven Programming

   The web application is made in a way so that it is event driven. This means that the flow of the program execution is determined on the user’s actions. So, if the users redirect to a new page then the backend will route the website to the new page and display the web contents. Just like that the data is only visualized if the user selects the button to visualize it which will cause an event.

Introduction

The main problem that our team encountered with today's dataset/data portals was that if you need to see the data you would need to download it and apply some data visualization tool on it to actually see the data. With our new feature, users can directly get access to the files and in addition to that they are able to view the data right there on spot without having to download it. The main reason was that when browsing the data portal for another idea that we had, we faced a big issue of having to repeatedly download datasets and then seeing if they are applicable for us or not. And with this we came up with another idea and that was WinData. With this motivation we came up with a plan to eliminate that redundancy and increase the usability of the data portals by simply having just one button that visualize the data for the user right there on the spot. We discussed the various tools and languages that will be required to build such an application. We decided on using HTML, CSS, and Bootstrap for the front end of the website, this is the content that the users will be able to see and Bootstrap will be used to have a better UX (user experience). As for the backend the team decided to move forward with Python and its micro-web framework Flask. This will be used to connect the data visualization script with the front end, and it will also ensure proper routing for the website. For the data visualization we went with Matplotlib and Pandas for data manipulation. This report will cover multiple things that are related to the project such as some related work that has been done before by others. What approach we used, how did we build the application and what testing did we use. In the end we would discuss if the project turned out how we intended to be, were there any bugs or issues that the team faced. It will also cover some of the future plans for WinData and how it will continue to improve.

Related Work

Before starting the project, it was important to ensure that such kind of technology or service does not exist already. In order to ensure this, we went through multiple city's data portals such as Windsor's, Toronto's and many other. And we found the common issue was that the need for downloading the data. With this we moved forward with out idea and though about all the new features we can add that will set us apart from the others. We decided to focus firstly on the data visualization feature as this was the main selling point of our application. Due to this we researched what tools we can use and how it can be implemented. By the first iteration not much work had been done but after that iteration the team picked up its pace and by the end of the timeline we had a fully functioning website that could display data on the go.

Approach

With having MVC as the main architectural pattern, the team decided to use the five major design patterns to make the code overall efficiency and not redundant. The GRASP pattern was used such as: Creator, Expert, Low Coupling, Controller, High Cohesion. To make it easy diagrams will be attached in the Appendix A for visual aid.

- 1.0: VisualizeData is the creator class that gets the data from the .csv or .xls file from the project. It creates an instance of the Database class which retrieves the dataset and is then loaded up for data manipulation by a library called Pandas.
- 1.1: This is a sequence diagram which represents the usual flow of the website and how the user will interact with the website. This sequence diagram describes the user loading onto the website and click the Visualize button for their chosen dataset.
- 1.2: This is the use case diagram; the purpose of this diagram is to show the use cases and the expected behaviour that is supposed to happen in the software. This use case shows the main cases that can occur and what the output/result should be.
- Our application has a good structure in terms of low dependency. All the classes interact with each other through functions and they do not depend on each other. This means that the classes are made with the idea that one class for one job and it does that job perfectly. For example the data visualization class is not in charge of retrieving the data, it uses another class to retrieve that data and then it performs the data visualization along with data manipulation.
- Our application also maintains low coupling. Low coupling is how much do the different modules rely on each other. The modules should be as independent as possible so when changing one module it does not affect other modules.
- We also ensured that it follows High Cohesion, this means that the elements within a particular module are related to the functionality of that module. This means designing a module to do one specific job and to have it perform it well.

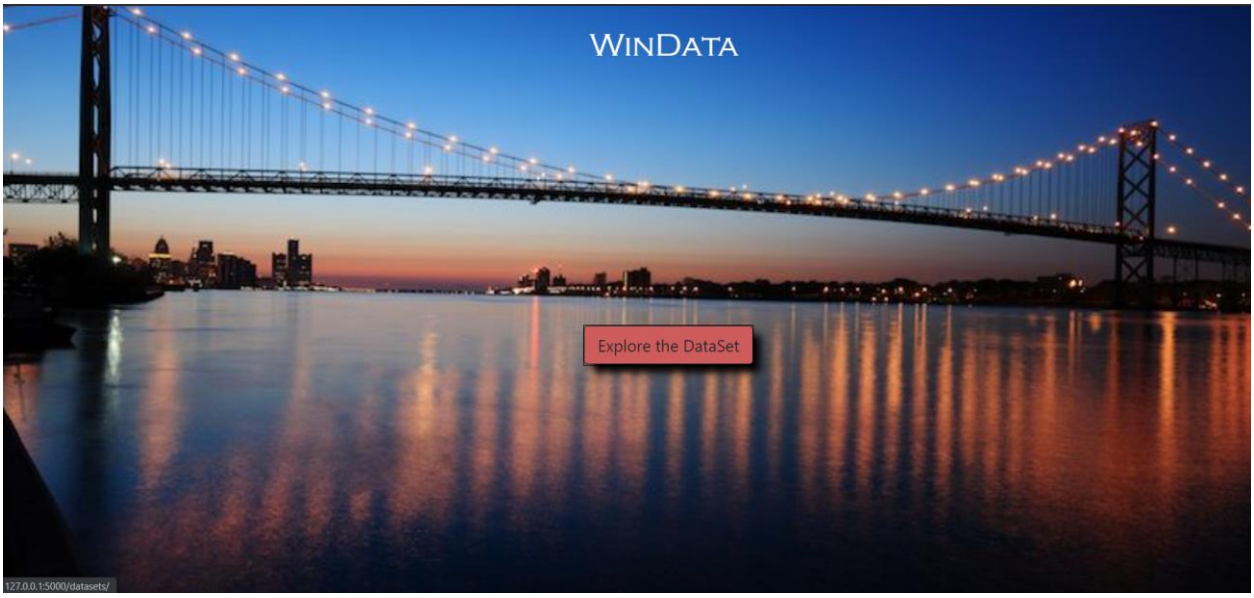
Experimental setup

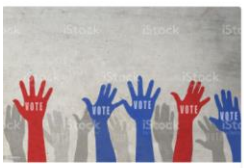
In the Inception phase, our team came up many features that we could implement. Some of them were filtering search for datasets by specific keywords, creating an API for the data portal, and visualizing datasets. After a while we decided to put our focus on the visualization of the dataset as this was the feature that many data portals did not have. Having this feature in data portals would increase the usability, the efficiency of the data portal overall. All the user would have to do is click on the button “Visualize Dataset” and this would open up a graph of the chosen dataset.

Then during out Elaboration Iteration, I phase, we realized that the main issue we faced was the dataset given to us from the data portal was not ready to be visualized right away. A bit o data manipulation was required. For example, the elections2010 dataset had the candidates and their respective votes. The issue was that in the .xls file, it was created with multiple districts in it. So, it meant that the candidates in the column “Candidate” occurred multiple times for each new district and their votes for that district only. When we tried to graph this using Pandas and Matplotlib, the graph was very wrong since it graphed the same candidate multiple times for each new district. This was not what the team was looking for. An example of this is attached in the appendix A 1.3. We tested out code with other datasets that had similar type of issue and the result was the same for all of them. From all of this team decided that the best option is not to just get the data from the data portal and graphing it but rather getting the data and then applying data manipulation to the data frame created by Pandas and then displaying that data. Another issue we found was the frontend was not very user friendly, due to this it was hard to navigate the website in the beginning. An example of this is in the appendix A 1.4.

So then in elaboration iteration II, we came up with a new plan that was better than the last one. Our team decided to use Pandas for data manipulation and matplotlib for graphing the data with only key row and column and a specific type of graph for that specific dataset For example a dataset regarding Elections, a bar graph would be more suitable than other graph such as a line graph. With the new library Pandas, we were able to resolve all the previous issues that we had with the data. This allowed us to create a good and concise data visual that could be used by the users. We also used a web micro-framework called Flask which allowed us to connect the

frontend to the backend and the ability to display the data when the user wants it to. We also fixed up the website by using a proper structure and layout to the overall website. We created a introduction/landing page and then they would proceed to a page that contains all the dataset and from that they are able to open up a specific dataset and then in that page they can view all the files that they can download, the details regarding the dataset and the ability to visualize the data. Below is an example for the website:





Official Election 2010

This dataset is about the official elections from 2010. You will find the votes and runners in the election

Go to Dataset

Arena

These point files contain the Name, Address, URL and X & Y Coordinates of each of the City's owned and operated arenas.

Go to Dataset

Hospital

These point files contain the name, address and X&Y Coordinates of each of the City's hospitals.

Go to Dataset

WinData Catalogue

Elections results 2010

Data Custodian:

Manager, Records & Elections.

Data Currency Comments:

The files were posted on December 2010.

Dataset Description:

These point file contain information for the Candidate Name, Total for that specific Office.

Data Accuracy Comments:

City of Windsor Municipal Election 2010 Data Official Results.

Attributes:

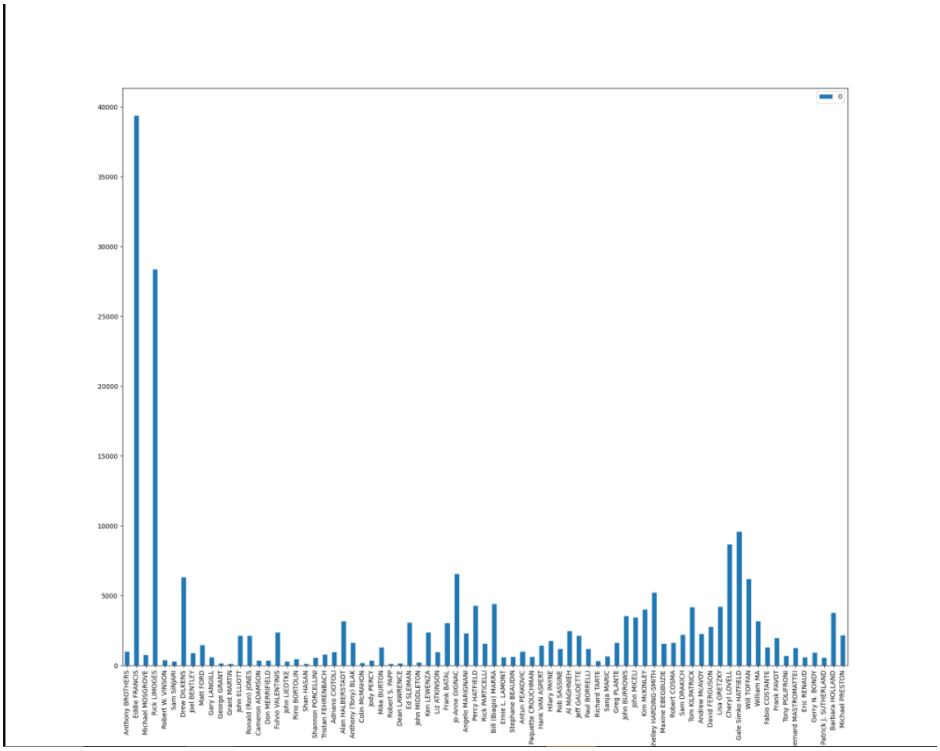
The attributes include the Ward, Poll, Poll Name, Office Title, Candidate, Total, and Notes.

Relevant Downloads:

Election2010.xls

Go Back | Visualize Data

With the new website, it gave the application a nice new modern fresh look. This would help the users navigate the website which would increase the UI and UX of the website. The final data visualization of the elections results 2010 looks like the below:



Discussion

Initially we planned on implemented many more features such as: filtering search for datasets by specific keywords, creating an API for the data portal, and visualizing datasets**.** We decided to follow the MVC structure and this resulted in the project turning out better than expected. It gave the project more structure which in the end resulted in a good way. The result was excellent and much better than what we had hoped for. With our focus and our business plan being:

> *Residents of City of Windsor can use this for access open data for research, application development or to improve their interaction with City of Windsor's services and facilities. The goal is to make the access as easy and convenient as possible.*

This goal was achieved in the end, we successfully managed to create a new data portal that was much better in terms of useability of the data. We managed to create an easier version of the pre-existing data portals. With a click of the button the user could view the data, of their chosen dataset. Along with this other click of the mouse allowed them to directly download the files they need if they wish to use it. With the use of HTML, CSS, and Bootstrap, we were also able to create the nice, appealing, use to easy website that attracted the users. This ensured that users and their experience was met. Not a lot of code was used form Elaboration Iteration I due to the poor design and structure of the code, we had to restructure the whole website and the backend as well. In the end this was a good choice on our behalf since it resulted in a better well-maintained website. For the testing phase the team tested with multiple different datasets and check if the backend was working correctly of not. A rigours testing was done by using different data types and datasets to make sure that the data visualization was working just the way it was intended to. Overall, the team was happy with their final product, and how it turned out. WinData is the new way to access open data. We managed to successfully redefine what data portals meant.

Conclusion and Future Work

Our main object for this project was to redefine what and how to use data portals. Our goal was to create a new type of data portal that not only contained all the key features that all pre-existing data portals have but to have one extra one and that is the ability to visualize the dataset without having the need to download the dataset. We successfully managed to implement this core feature and the users are able to visualize the dataset. We were able to mange this by using some technologies such as Flask, Pandas, Matplotlib. Some of the thing we failed to achieve was adding the other additional features we planned on implementing. Those features were creating an API for the data portal and the ability to filter out datasets with specific key words. For example, the key word “elections” would only show datasets that are related to elections. The assumption that we make our application is when we visualize the dataset, we specifically choose the key attributes that we think should be graphed/displayed. For example, the elections dataset we assumed that the user would most likely be interested in seeing overall votes the candidates received from all the districts combined. This could also be a limitation, another limitation would be the variety of dataset available, as of now WinData does not have that many datasets to offer, since it is still in its development phase. The main benefit WinData has over other data portals is the ability to display the data without them having to download the dataset or the data.

For the future iterations, our team plans on doing some important things. One of them is hosting the website so users from all around the world can access it and use the website. Second one is implementing the search feature, this would help users more with the data browsing and overall making the website easier to use. We then plan on making an API for developers around the world so they can use the data for their scientific research. Another major thing we would want to add is more datasets as of now WinData does not have many datasets.

References

The following references are made using IEEE formatting.

Tutorialspoint.com. 2020. MVC Framework - Introduction - Tutorialspoint. [online] Available at: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm [Accessed 30 October 2020]

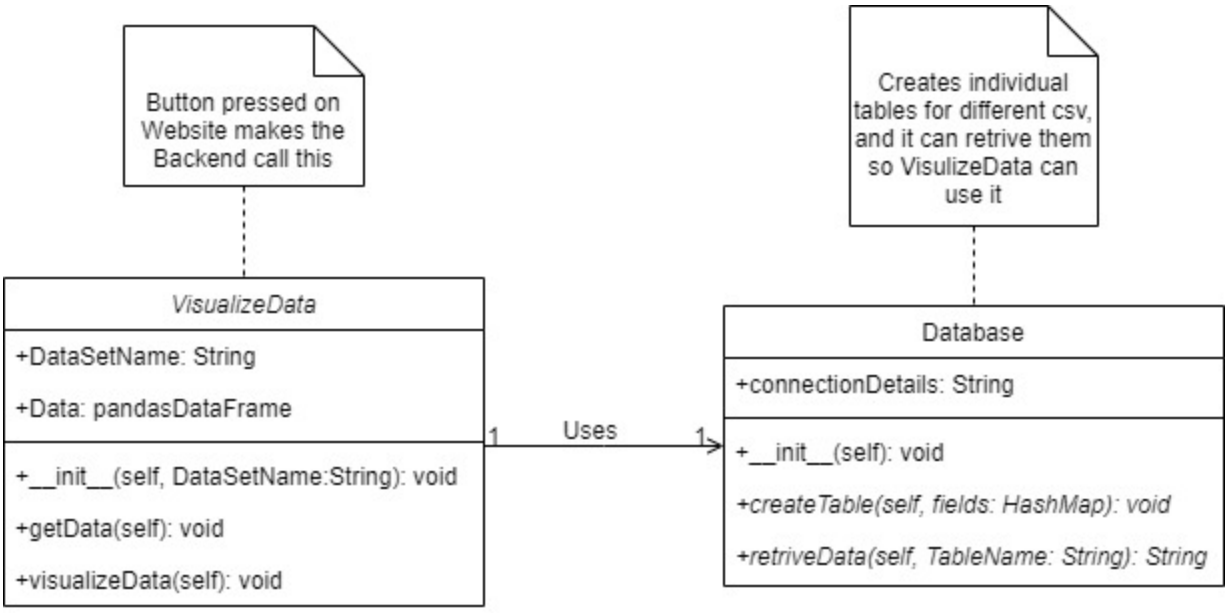
Montiel, I., 2020. Low Coupling, High Cohesion. [online] Medium. Available at: https://medium.com/clarityhub/low-coupling-high-cohesion-3610e35ac4a6 [Accessed 1 November 2020].

Tutorialspoint.com. 2020. Flask Tutorial - Tutorialspoint. [online] Available at: https://www.tutorialspoint.com/flask/index.htm [Accessed 13 November 2020].

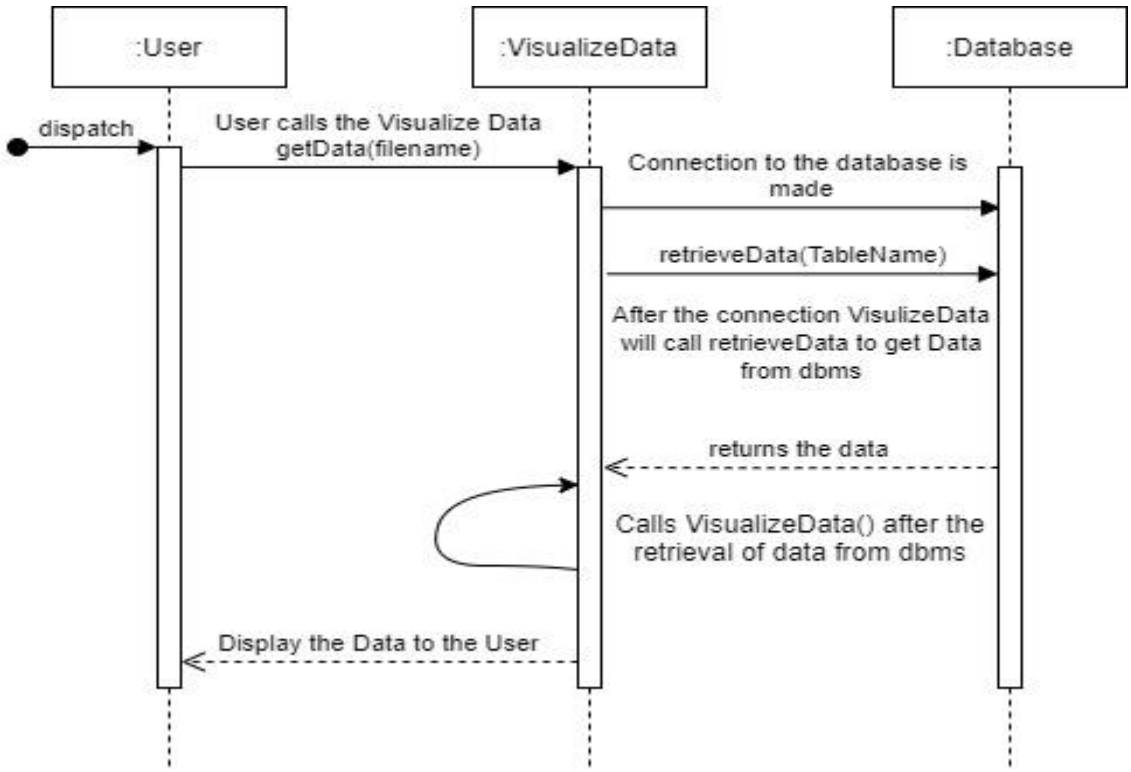
McIntire, G., Washington, L. and Martin, B., 2020. Python Pandas Tutorial: A Complete Introduction For Beginners. [online] Learndatasci.com. Available at: https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/ [Accessed 14 November 2020].

Appendix A

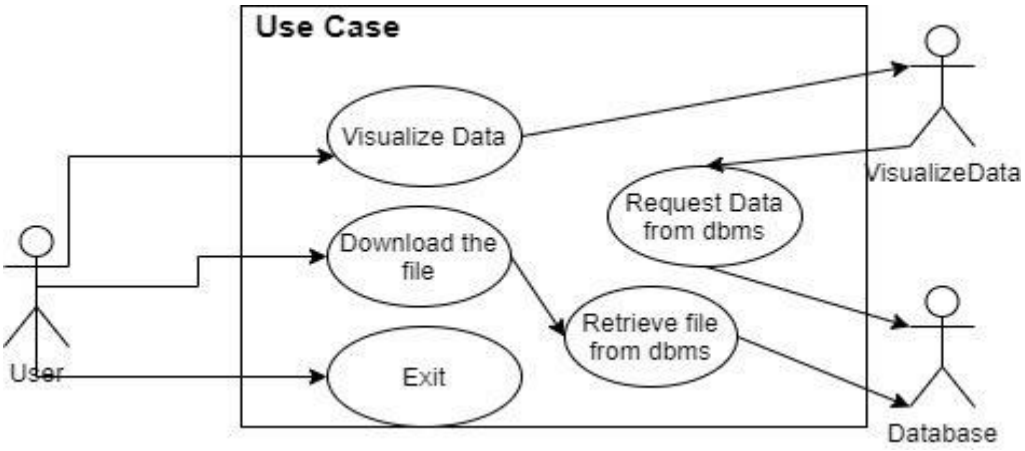
1.0



1.1



1.2


Use Case
Visualize Data
Download the file
Exit
Request Data from dbms
Retrieve file from dbms
User
VisualizeData
Database

1.3

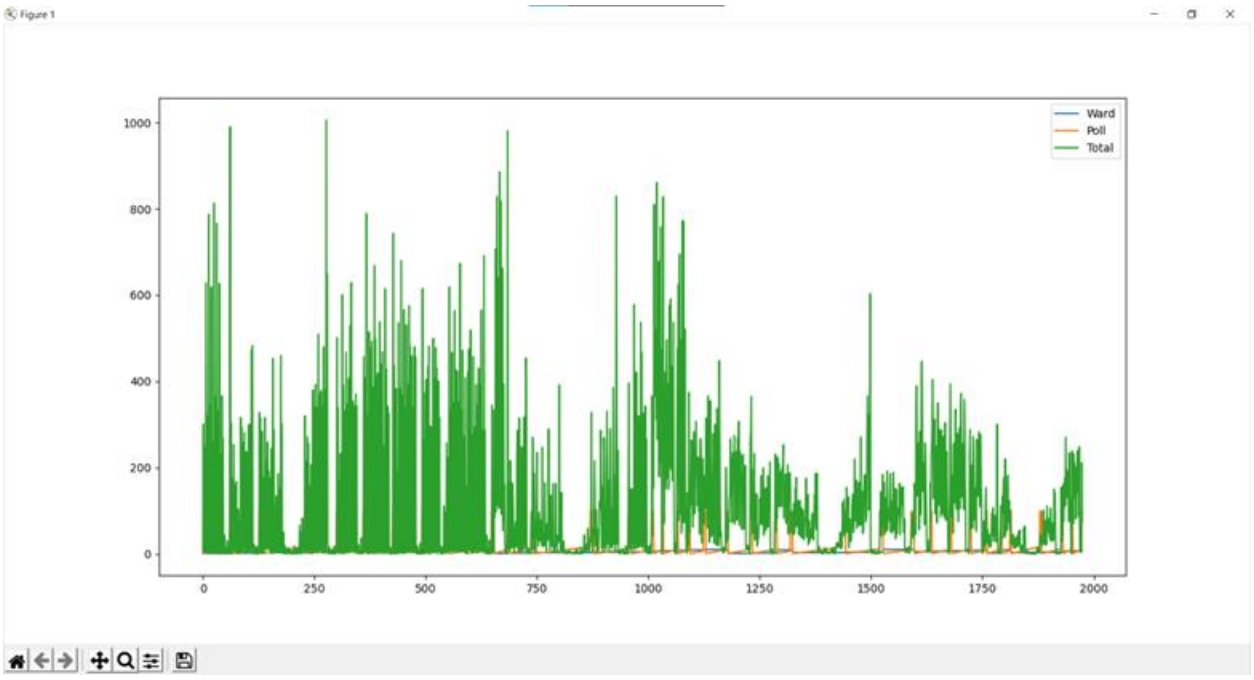


Figure 1
Ward
Poll
Total
1000
800
600
400
200
0
0
250
500
750
1000
1250
1500
1750
2000

1.4