**PlotMe**

Interactive Data Visualizing Application

**Design Specification Document**

Friday, April 10, 2015

Computer Science – Comp Sci 2XB3

Group 18

Saim Malik, Ahmed Khan and Zayan Imtiaz

Team Roles

|  |  |  |
| --- | --- | --- |
| **Name** | **Student Number** | **Roles** |
| Ahmed Khan |  | Project Leader, Programmer, Back-end, Web Development |
| Zayan Imtiaz | 1152665 | Researcher, Programmer, Log Admin, Back-end |
| Saim Malik | 1321796 | Programmer, Tested, Front-end, UI, Web Development and Design |

“*By virtue of submitting this document we electronically sign and date that the work being submitted by all the individuals in the group is their exclusive work as a group and we consent to make available the application developed through [CS] or [SE]-2XB3 project, the reports, presentations, and assignments (not including my name and student number) for future teaching purposes.”*

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Role(s)** | **Contributions** | **Comments** |
| Saim | Web Design | Website’s user interface | Designed the UI for all the web pages using HTML5, CSS (Bootstrap), JavaScript (JQuery) |
| Ahmed | Java Programmer | Implemented sorting algorithms | Created an optimized sorting algorithm that uses both merge and insertion sort based on the size of the data |
| Zayan | Java Programmer | CSV file parser | Wrote a parser that extracts data from a given CSV file and returns it in a format usable by the program |
| Ahmed | Web Developer | Graphs using d3js | Made visually appealing graphs using d3js library and JavaScript |
| Saim | Web Developer | Wrote the controller | Combined the back-end of the application with the front-end using the Grails Framework (and MVC design) |
| Zayan | Documentation | Requirements and Specification Document | Wrote the Requirements and Specification Document |
| Saim | Documentation | Design Specification Document | Wrote the Design Specification Document |
| Saim | Documentation | Presentation Slides | Made PowerPoint presentation slides |
| Zayan | Log Admin | Group log file | Updated the group’s progress in the log file |
| Ahmed | Java Programmer | Implemented analyze functionality | Programmed an algorithm to analyze a given dataset and present the ‘outliers’ in the data in graph format |
| Zayan | Java Programmer | Implemented search algorithms | Designed a search algorithm that users a BST to efficiently search through the dataset for a given input |
| Saim | Tester | Tested the application | Ran the application with several input files and different graph settings to assure the robustness of the application |

**Abstract**

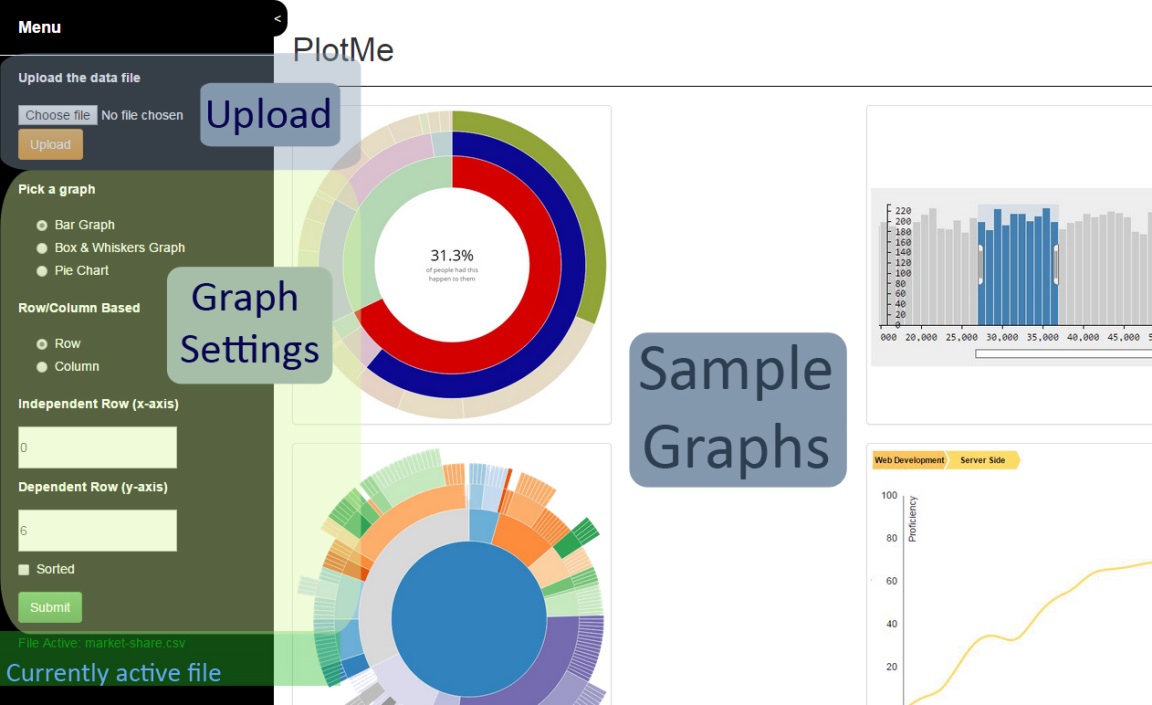
PlotMe is a web-based application that allows its users to visualize data files as interactive graphs. The application allows the user to choose from several graph settings to account for the structure the data could be in. For instance, some data sets have headings in the first row and others might have headings in the first column; the graph settings in this application account for both such conditions and several others. Aside from the functionality, PlotMe has an interface that is both visually appealing and easy to use. The graphs generated include Bar Graph, Box and Whiskers Graph and Pie Chart.

**Table of Contents**

* User Interface Guide (How to use)
* Class/Modules Description
  + Public Entities
  + Private Entities
* Use Relationship (UML)
* Evaluation of Design

User Interface Guide (How to Use)

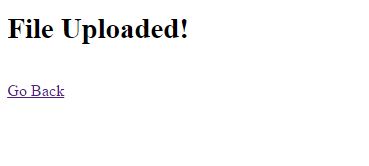
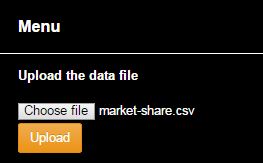
Open a web browser and go to <http://plotme.cfapps.io> (or <http://plotme.cfapps.io/main> if the first link doesn’t work)



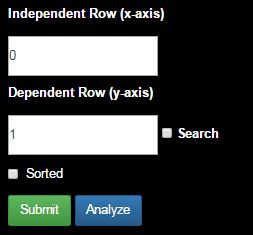
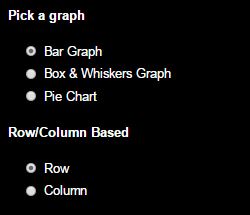
Click on the pictures of the sample graphs to view them (most of them are interactive – i.e. you can click on the graph and it will do something)

To upload your own dataset to be graphed, expand the collapsible menu (if it isn’t already) and follow these steps:

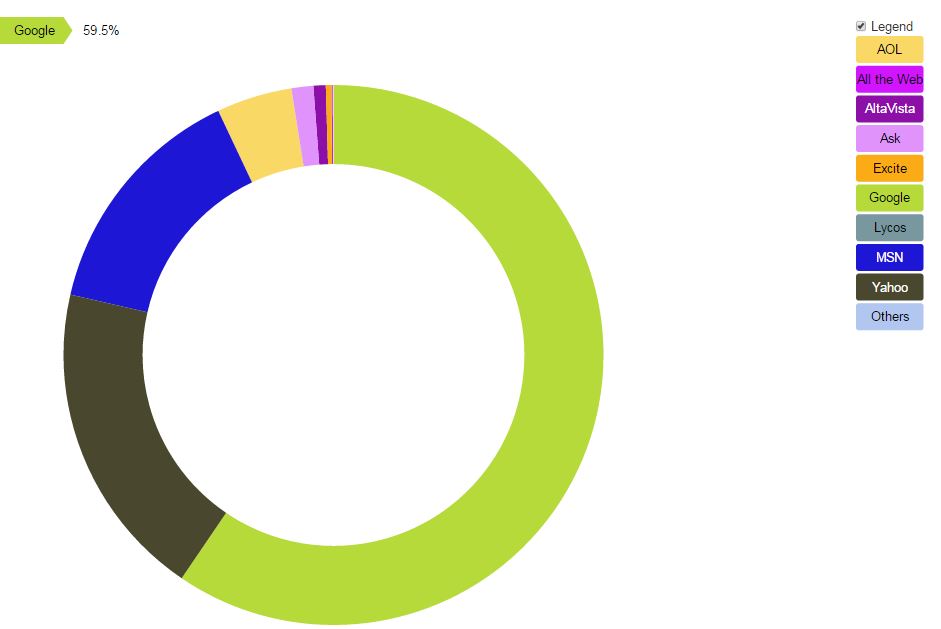
* Click Choose File
* Go to the “sample-data” folder (provided with the submission) and pick one of the CSV files
* Click Upload; it should open a new window that says “File Uploaded” with a “Go Back” link



* Click “Go Back”; the sidebar menu should now have different settings visible to graph the data
* Pick one of the three available graphs (Bar Graph, Box and Whiskers or Pie Chart)
* Choose whether the dataset is “Row Based” or “Column Based”
  + Row Based datasets are those in which the headings are the **first** **row** and **each row after** that has values with respect to its appropriate column
  + Column Based datasets are those in which the headings are the **first** **column** and **each column after** that has values with respect to its appropriate row



* In the first textbox, write the **row (if Row Based)/column (if Column Based) number** for the x-axis (usually the first row/column with all the headings) – **NOTE**: The numbers start with 0; so for the first row/column, input 0; for the second row/column, input 1, etc.
* In the second textbox, you have two options:
  + Search is NOT checked
    - Write the **row (if Row Based)/column (if Column Based) number** with the data for the y-axis (can’t be the same as the x-axis)
  + Search is checked
    - Write the **column number,<search input> (if Row Based)/row number,<search input> (if Column Based)** and the program will automatically find the row/column with the searched text (an example for the “market-share.csv” dataset (a Row Based dataset) would be, “0,1/1/2005” – where “0” is the “Date” column and the “1/1/2005” is the Date being searched for)
* Check the Sorted checkbox if you want the graphed data to be sorted
* Finally, click the Submit button (that should now be enabled)
* A new window should open with a graph based on the provided settings
* To change any settings or to upload a new dataset, “Go Back” and follow the same process as described above

*Pie Chart for market-share.csv dataset*

Classes/Modules Description

**Parser.groovy**

This class is used to parse the given data sets and store the results in terms of points containing independent variables (x-axis) and dependent variables (y-axis).

**Point.groovy**

This class creates an object Point which contains the independent variable, x axis stored as a String and the dependent variable y axis stored as a double. Storing it in this method makes access of required pairs convenient.

**Data.groovy**

This Data class is used to store the ArrayList containing the pairs of points for the independent and dependent variables as well as other relevant information. This is converted later into a Json object for the d3 library.

**RedBlackBST.java**

This class was taken from the 2XB3/2CO3 class textbook website:  
http://algs4.cs.princeton.edu/33balanced/RedBlackBST.java.html  
  
It is used to search through a given row or column in an ArrayList containing a parsed dataset (with rows corresponding to lines) and to return the corresponding column or row values. For these purposes only the put and get methods were utilized.

**Public Entities**

**Parser.groovy**

**public static ArrayList<**String[]**> getArray(**String S**)**

Input: Name of dataset csv file to be parsed taken as a string S.   
Output: Returns an ArrayList containing rows of all the lines in the dataset as a list of strings.

**public static Data getPairCol(**ArrayList<String[]**>** s, int x, int y**)**

Input:

An ArrayList containing rows of all the lines in the dataset as a list of strings s.

The independent column (x) and dependent column (y)

Output:  
Returns a Data object containing an ArrayList which includes the list of points with each point containing an independent variable (String) and dependent variable (double).

**public static Data getPairRow(**ArrayList<String[]> s, int x, int y**)**

Input:

An ArrayList containing rows of all the lines in the dataset as a list of strings s.

The independent row (x) and dependent row (y)

Output:  
Returns a Data object containing an ArrayList which includes the list of points with each point containing an independent variable (String) and dependent variable (double).

**public static Data getPair(**ArrayList<String[]> s, int x, int y, String u**)**

Input:  
An ArrayList containing rows of all the lines in the dataset as a list of strings s. The independent row (x) and dependent row (y). A String u which specifies is the user wants to create a pair using rows (“row”) or columns (“col”).

Output:

Returns a Data object containing an ArrayList which includes the list of points with each point containing an independent variable (String) and dependent variable (double).

\*This method works by calling either getPairCol or getPairRow depending on what is specified in String u.

**public static Data getSetofPairs (**ArrayList<String[]> s, int x, int y, String u**)**

Input:

An ArrayList containing rows of all the lines in the dataset as a list of strings s. The independent row (x) and dependent row (y). A String u which specifies is the user wants to create a pair using rows (“row”) or columns (“col”).

Output:

Returns a Data object containing an ArrayList which includes the list of points that include all possible dependent variables for a given independent variable with each point containing an independent variable (String) and dependent variable (double).

**public static RedBlackBST <**String, Integer**> getBSTRow (**ArrayList<String[]> a, int x**)**

Input:

An ArrayList containing rows of all the lines in the dataset as a list of strings a.

The row to be stored as an int x.   
  
Output:

A RedBackBST that contains the row elements as strings and their corresponding column numbers.

\*This is utilized in the UI for the user's convenience so they can can search for a column number by just knowing the String they desire as opposed to the column number where it's located.

**public static RedBlackBST <String, Integer> getBSTCol (ArrayList<String[]> a, int x)**

Input:

An ArrayList containing rows of all the lines in the dataset as a list of strings a.

The column to be stored as an int x.   
  
Output:

A RedBackBST that contains the column elements as strings and their corresponding row numbers.

\*This is utilized in the UI for the user's convenience so they can can search for a row number by just knowing the String they desire as opposed to the row number where it's located.

**public static RedBlackBST <**String, Integer**> getBST (**ArrayList<String[]> a, int x, String okay**)**

Input:

An ArrayList containing rows of all the lines in the dataset as a list of strings a.

The column or row to be stored as an int x.

A String Okay containing “row” or “col” to determine which is being stored.

Output:

A RedBackBST that contains the column/row elements as strings and their corresponding row/column numbers.

\*this function calls either getBSTCol or getBSTRow to create the BST depending on what the user specifies for string Okay.

**Point.groovy**

**public Point (**String n, double v**)**

Constructor that initializes name and value

**public void setName (**String a**)**

Input: String containing name

Output: sets Name

**public String getName ()**

Output: returns name

**public void setValue (**Double a**)**

Input: Double containing value

Output: sets Value

**public double getValue ()**

Output: returns Value

@Override **public String toString()**{

Output: Prints out current objects of class, used for testing purposes

**Data.groovy**

**public Data (**String N, String x, String y, ArrayList<Point> d**)**

Constructor used to initialize the Data object.

**public String getName()**

Output: return Name;

**public String getXaxis ()**

Output: return x-axis;

**public String getYaxis ()**

Output: return y-axis;

**public ArrayList<**Point**> getData ()**

Output: return data;

**RedBlackBST.java**

**public void put(**Key key, Value val**)**

Input: Key (String name) and value (corresponding row or column number) to be stored.

**public Value get(**Key key**)**

Input: The string name (key) one is searching for.

Output: Return the corresponding row or column number.

**Private Entities**

**Point.groovy**

This class creates an object Point which contains the independent variable, x axis stored as a String and the dependent variable y axis stored as a double. Storing it in this method makes access of required pairs convenient.

**private String name;**

independent instance variable

**private double value;**

dependent instance variable

**Data.groovy**

**private String Name;**

can be used to store the String name

**private String xaxis;**

Stores the name of the x axis

**private String yaxis;**

stores the name of the y axis

**private ArrayList<**Point**> data;**

Stores an ArrayList of point objects for the independent and dependent value pairs.