# Project Report

## Group 12

**Members:**
Nicky Casey, Wessam Essam S Gholam, Christine Hallahan, Judy Meriel Dick, Eoin Corcoran.

**Introduction:**
The aim of this project was to design an application which would analyse data from taxis in New York City using Processing. The data includes the taxis' individual identification numbers, the date and distance of each journey and the pickup and drop-off locations and times of each trip.

We had to work as part of a team to create this application and design alternative ways to display different aspects of the data. The Git repository infrastructure allowed us to share our work and collaborate on the program.

**Design:**
Buffered Reader:
After some attempts trying to read in the data files, we decided to use the buffered reader which allowed us to read all the data from both of the big files. Instead of loading the whole file into a string, the buffered reader reads it line by line.

Data Extraction:
The buffered reader is implemented in the data extraction class. Also included in this class is our different queries. The queries use the line that the buffered reader is reading and manipulates it to store information in different arrays. These arrays are declared in the main which allows us to use them in the different charts.

Screens:
A screen class was created and multiple screens are implemented using this. It allowed the home screen and different screens for each section of the program to be easily designed . We have a data type value *'currentScreen'* which allows the program to keep track of which screen it is on and which screen to switch to.

Widgets:
We designed a widget class to allow numerous widgets to be used throughout the application. For the widgets on the main screen and the buttons on each screen to return to the main screen, we use a picture of a cartoon taxi with the name of the page each widget goes to. We also use widgets to flick between queries for the bar chart and pie chart, and the date picker for the map avails of this class as well.

Bar Chart:

The aim of implementing bar charts was to illustrate data in a manner that was easy to see and understand. The bar chart class takes in an array of floats, which is the numeric data taken from the files and an array of strings which are the labels on the x-axis of the bar chart. Also taken are the names of the x and y axis as strings and the position to place the bar chart on the screen. First the highest numeric value taken is determined, for example the highest average distance travelled. Then this highest number is divided by a number, and this highest number is continuously subtracted by the new number. The results of these operations are the numbers to be put on the y-axis. For example, 20/10 = 2. So the numbers on the y-axis are 20,18,16…0.

The margin and bar charts themselves are drawn with processing's "rect()" function. The positioning and size of the bars are determined by the amount of variables needed in the bar chart. The size of the bars scale to fit the chart as it is determined from operations on variables that are defined in the constructor.

Pie Chart:

We created a new pie chart class that would work for our queries as they could use floats. Hard coded values were used initially to test it but the finished result takes in a float array which will be the sectors on the pie chart, a string array which will be used to make the key and finally just a string which will be used for the title. These are passed in and drawn depending on which widget was pressed on the screen by the user. Then to get ready to draw the pie chart there are multiple for loops to carry out methods to get the total, the fraction of pie, then the degree the angle will be at to calculate that for each element in the float array. Different arcs are drawn, the next one starting where the other one finishes so they are all connected which will form a pie chart rather than drawing the pie chart then filling in the arcs later. Finally a key is created, which uses the elements in the string array and prints the corresponding color for that element in the pie chart as a rectangle and writes the value beside it every time it goes around the for loop.

Map:

This was our last visualisation in the program. It takes the required date from the user using the date picker, and drops pins on the map at various loactions. The map was created using an external library called "Unfolding maps", which is a processing library. The library provides the visualisation of maps, and allows pins to be dropped at certain coordinates.

Date Picker:

The Date Picker is implemented as an object. The user selects their desired date, which is saved in memory and can be changed once the mouse is pressed and a different date is chosen. Each time a new date is picked, the program takes the date, filters the data file, then all existing drop pins are removed from the map, and the new drop pins are placed again.

Heat Map:
The heat map basically shows the congested areas for both the pickup and drop-off points on a certain date in New York city. The algorithm used to implement the map is to make a grid of circles, where each circle must intersect with the surrounding circles in order to have a more accurate result. The program counts each pin for both pickups and drop-offs in a certain circle, and colours each circle accordingly. The colour gets darker or lighter depending on the number of pins. Each taxi trip is linked by a line. For example, the location of the pickup is linked by a line to the drop off location.

Satellite Map:
The Satellite map has the pins of taxi pickup and drop-off locations. The “Unfolding maps” library provides different kinds of maps, and the Satellite Map is one of them. This map was implemented in our program because it shows streets and buildings, which we thought would be an important feature especially when looking closer at the taxi pickups and drop off locations.

**Team:**
We organised to meet up once each week outside the allocated lab times. We used this time to discuss the progress of the program and what was needed to be included for that week’s lab. We divided up any work that was needed to be completed and discussed any problems that were encountered.

We kept in contact throughout the process by a group chat where we could ask questions, help each other with problems and organise meetings.

We split up the work as shown in the table below:

| | |
|---|---|
| **Nicky** | Class to represent multiple queries in bar chart form |
| **Wessam** | Create maps to display taxi locations on specific dates |
| **Christine** | Implement a screen class to allow for multiple screens and a widget class to enable the user to interact with the program |
| **Judy** | Read in the data files and extract relevant information |
| **Eoin** | Create a pie chart class to display different queries |

We generally stuck to these but we helped each other if a problem was encountered and we collaborated together on certain parts of the program.

**Features:**

- When the mouse is over certain widgets, a white border appears on that widget to highlight it.
- A click sound was added using an internal library called ‘minim’. The play() method of the library is called once the mouse is pressed.
- Bar charts and pie charts are used to show different queries. Multiple widgets enable the user to switch between these queries.
- The date picker allows the user to choose which day’s trips the map will show.
- The map contains two different colour pins which represent the pickup and drop-off points of all the taxis that day in New York City.
- A heat map is displayed to indicate which parts of the city had the most activity.

Here are some important keys to use the map:

- Press ‘o’ to zoom out
- Press ‘z’ to zoom in
- Press ‘1’ for heat map
- Press ‘2’ for satellite map
- Press ‘m’ to return to the main menu

**Problems:**

- We had initial problems with trying to read in the big data files. We found that with the loadStrings() method, it ran out of memory very quickly, which meant we would have had to reduce the size of the files in order to use this method. We then decided to use the buffered reader so we did not have to alter the data file in order to access the information.

- Finding the correct way to determine the size of the bars and positioning of the labels for the bar chart. This was solved through trial and error of different operations to calculate the correct sizes and positions. Changing the size of the chart to be quite small will make the charts labels appear on top of each other. This problem is addressed by having the size of the chart be unchanged.

- A pie chart class was initially created that drew a pie chart and then filed in the angles later. However this was a problem as when a pie chart was being drawn, it could only use an integer array where many of our queries would have decimal places.

**Conclusion:**

The application that we created as a team is able to read in the data files about taxis in New York City, extract data accordingly and display the information in various ways. It allows the user to interact with it and select how the data should be viewed and what queries to display.