**Team 31 – Minutes**

**Members**

Ahmed Abourezk

Oluwafirebami Kukoyi

Kevin Onwubuilo

Conor Kelly

Boyi Zhang

Ciaran Nolan

**Meeting 1 - *13/3***

* Brainstormed what our project may look like and set a few guidelines as to how we would achieve it.
* Decided how we would meet in future. Mostly in person on Tuesday and Wednesday. However, the idea of a few online meetings also came up which would be good for reviewing and analysing our code together.
* Assigned tasks to prepare for our 1st lab.

***Conor:*** Set up Git repo and initialize it with a few of the basic documents. Add in a shell of what our processing folder may look like (Decided as group)

***Ahmed:*** Setup our initial screen. Decide on a size to use and implement this as a constant which can easily be altered as we progress. Additionally, add a graphic or text to give some context to what this code may do.

***Boyi:*** Define a clearer structure to the project. What classes may we use and what should they do? What will some of our screens look like?

***Ciaran:***  Write the code to read in the data. Decided where this should be implemented and how it will be done. Print to screen the data to show that data is being read in successfully.

***Oluwafirebami & Kevin:*** We need a way to store this data that were reading in. Write the outline of a Flight class which will allow us to store each flight read in as an object. This object will contain all the information relating to a particular flight.

**Meeting 2 - *20/3***

Thought about what our project should look like and achieve this week then divided the tasks to each team member.

***Conor:*** Implemented the basic screen and widget classes to our program. Set up a query variable that will handle the user’s queries and gets changed on the press of a particular button. Created the outline of our Render class which will decide which graph to create, and draw based on its own query variable. The screen class currently calls the Render object to draw itself.

***Ahmed:*** Re-arranged the widgets and adjusted the screen. Implemented a background and a heading for the program. Created a sketch for what the program will look like as of week 2.

***Boyi:***

***Ciaran:***  Building on last week’s code which read in the data from the file, wrote a code which created the query (5 most busiest airports). This code reads the data and parses through only the 5 airports with the highest flights. It prints these 5 airports to the screen to demonstrate methods work.

***Oluwafirebami:*** Drew the bar chart for the Busiest airport data, used the sorted data of origin states done by another team member to draw the bar chart

***Kevin:***

**Week 3**

Brush up on last week's work and solve issues we encountered such as backgrounds and buttons deviating positions. Create at minimum three solid queries based on a more streamlined query system and create clear ways to gather the date for these queries.

***Conor:*** Streamlined the query system, making the render object draw it's graph based on the query it currently holds through a switch statement and made it so the buttons control what the current query is. Cleared the screen of unused buttons. Added a screenTracker variable to enable the screen class to identify what screen it's displaying and change some of its functionalities based on this (i.e., what background is drawn). Completed a second query which displays the top 5 states for cancelled flights in the form of a bar chart. I implemented a bar charts class that allows for a much easier creation of bar charts where data and labels are passed into it through a constructor, and you can then just ask the chart object to draw itself. Created a dataProcessing tab where we can store functions that perform operations on the data and segments it for use in our queries.

***Ahmed:*** Fixed an issue with the text that occurred when checking the busiest airport query and done additional changes to the screens

***Boyi:***

***Ciaran:*** building on last weeks code, amended slight errors with the files of where

data was stored in the program. Then created another query shortest flights duration.

This lists the 5 shortest flights from the origin and flight time. I created a barchart to

display this information

***Oluwafirebami:*** Implemented code to create object of type flights for each flight in the flights.csv file so it is easier to manipulate the data in order to make queries and draw visual representation of them. Polished and designed the bar chart made in week 2, added x and y labels and title

***Kevin:***

**Week 4**

Heading into our final week we each took it upon ourselves to try and add on to what we have already created individually. This includes different UI elements which will allow us to hit this week’s minimum requirement. Additionally, we will all work on bettering our program where we see fit as we go along.

***Conor:*** Extended the cancelled flights query to include an additional option for users to select specific airports that we have data on and get the amount of cancellations for that individual airport printed to the console, with the aim of getting it displayed on screen in the coming days. I made use of the controlP5 library to implement this and spent many hours making sense of its barebones API documentation.

Also began the process of bettering the documentation of my code following the guidelines we saw in this weeks lecture.

***Ahmed:*** Finalised the screen and began work on a transition for the display of the program. Began work the dropdown box. Implemented an “About” button screen which includes a basic description of who built the program and the group.

***Boyi:***

***Ciaran:***  Fixed errors with method for calculating shortest flight. Changed to calculate longest flight instead. Incorporated a switch statement to account for time differences. Created a draw function which now draws a pie chart as well as the bar chart from before. Data displayed clearly and neatly, however still some changes to be made.

***Oluwafirebami:***

***Kevin:***