Visualisation project

**Air crash analysis**

By: Siddharth Kumar Ravichandar (28961145)

# Introduction

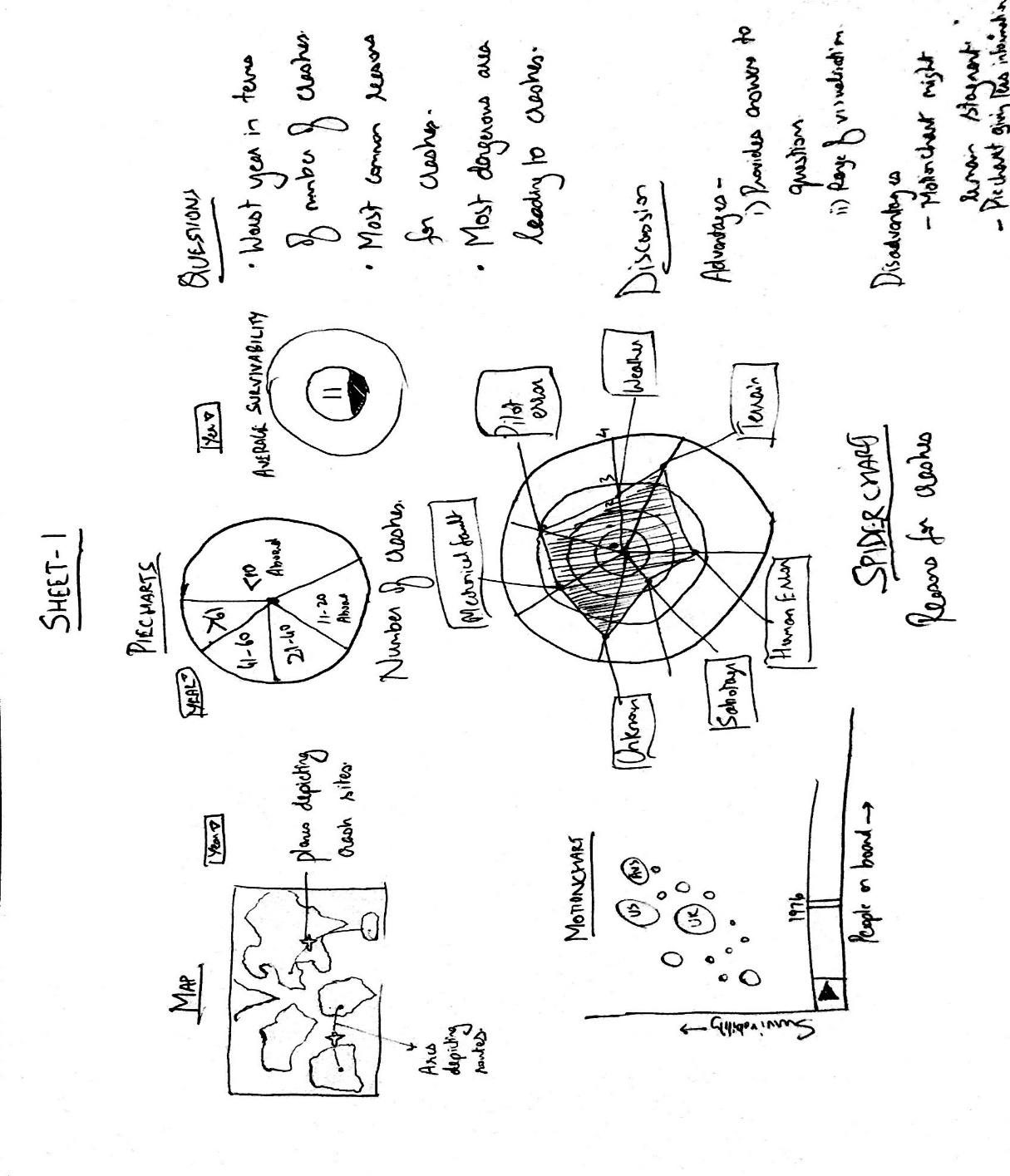
Over the course of a century after the invention of the plane, the number of crashes has only been rising with increased air traffic. After exploring the dataset in the exploration project, each crash over the last century was analysed and classified into various reasons. Through this visualisation project, I wanted to take it further and let the user explore the data more by providing appropriate tools to make the best out of the dataset chosen.

As I had mentioned in my Exploration project, my main motivation for choosing this topic was because I always wanted to be part of the aviation industry. But the fear of rising deaths due to airplane crashes, kept me from entering the industry. Hence this project is intended for people dreaming of entering the industry. This project would also prove to be vital for aircraft manufacturers by understanding the consequences of negligence in building faulty aircrafts. There are a lot of airlines these days that compromise the quality of the flight for passengers by bringing in budget flights. These obviously attract more customers. But the risks associated with it is far more than what little money is saved.

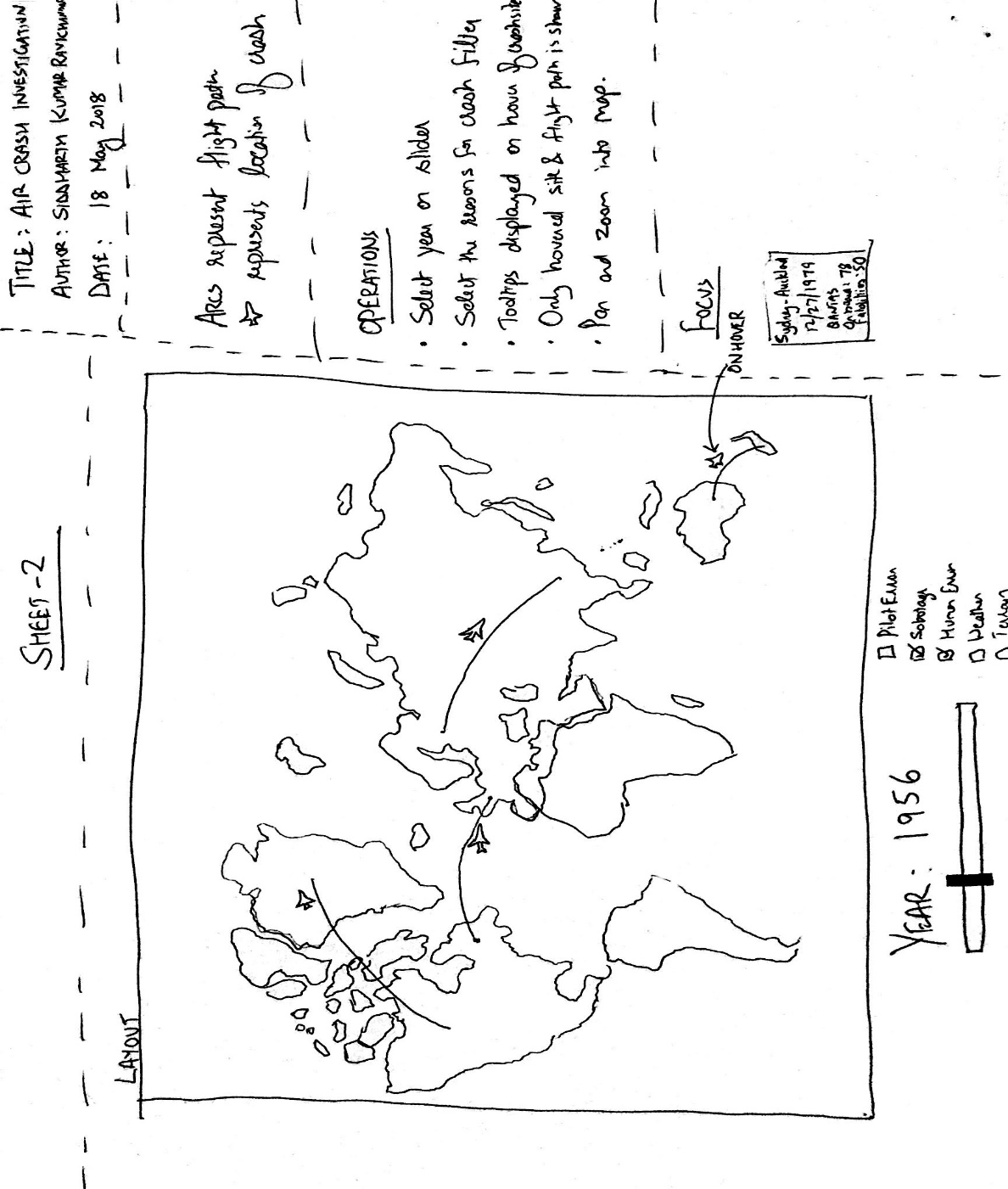
# Design

The way this project was carried out was by heavily making use of the 5-sheet design methodology. In this method, we first try brainstorming. The basic ideology here is to expand the world of possibilities. It is also kept in mind to focus as much as possible on quantity, which is to think of all designs that are possible.

Shown below is the first sheet of ideas that came across my mind while trying to think of possible ways to visualise all the air crashes there has been so far in the last century. Some of the ideas that were thought through were – map, motionchart, piecharts, spiderchart and liquid filling gauge.



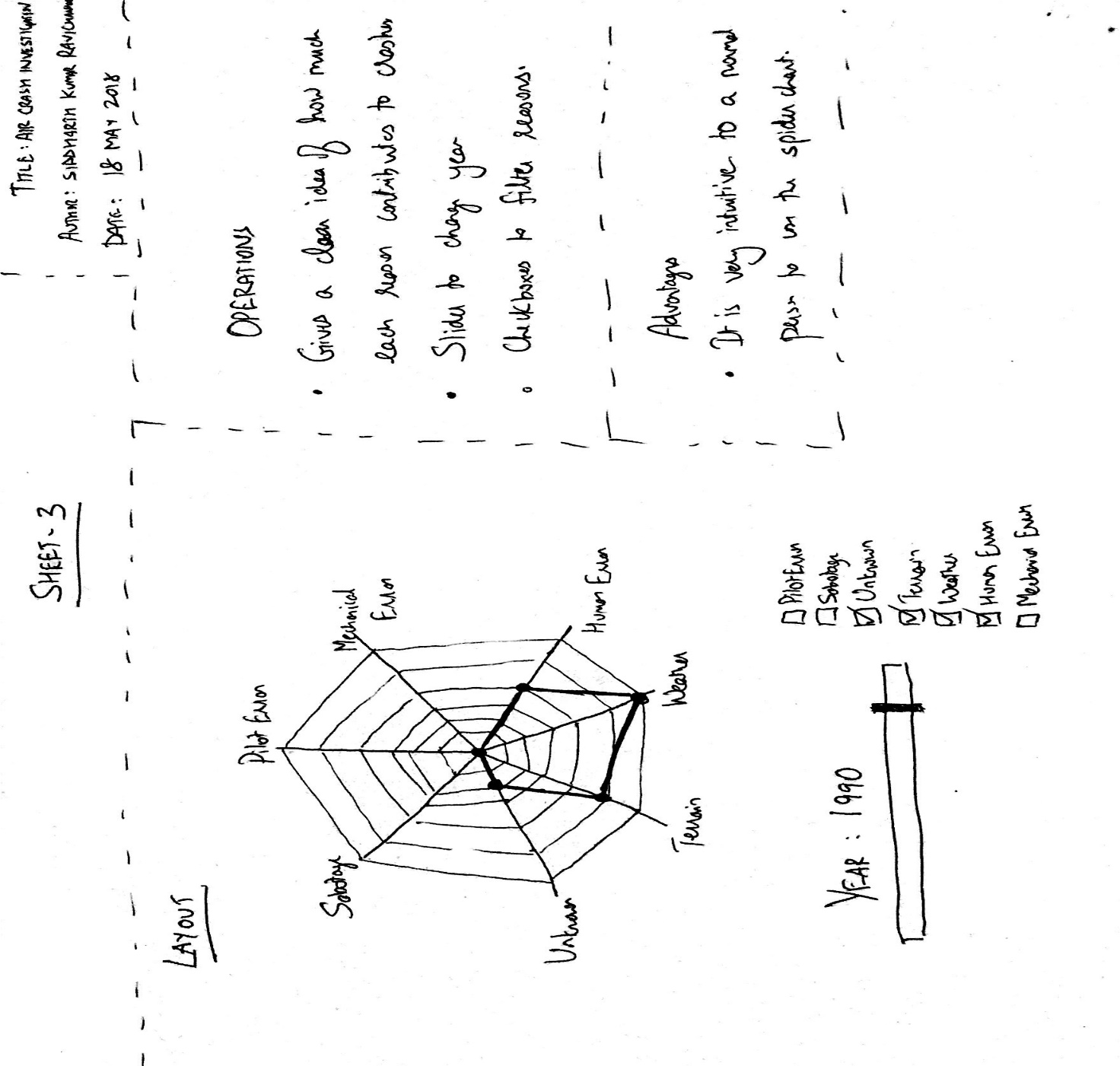
But not all ideas yielded successfully. By using the motionchart, I wanted to visualise the way number of people on board has affected the survivability chances given that you are involved in a crash. Motionchart could help us view this relation over the number of years for which we have data. But on trying to imagine the motionchart, there would not be much movement in the bubbles. So, it was decided to reject the idea of visualising using motionchart.



A map was a very obvious choice to show the routes that were taken by each flight that had crashed by using arcs. This map also shows the crash site, pin pointing the exact geocode of the crash.

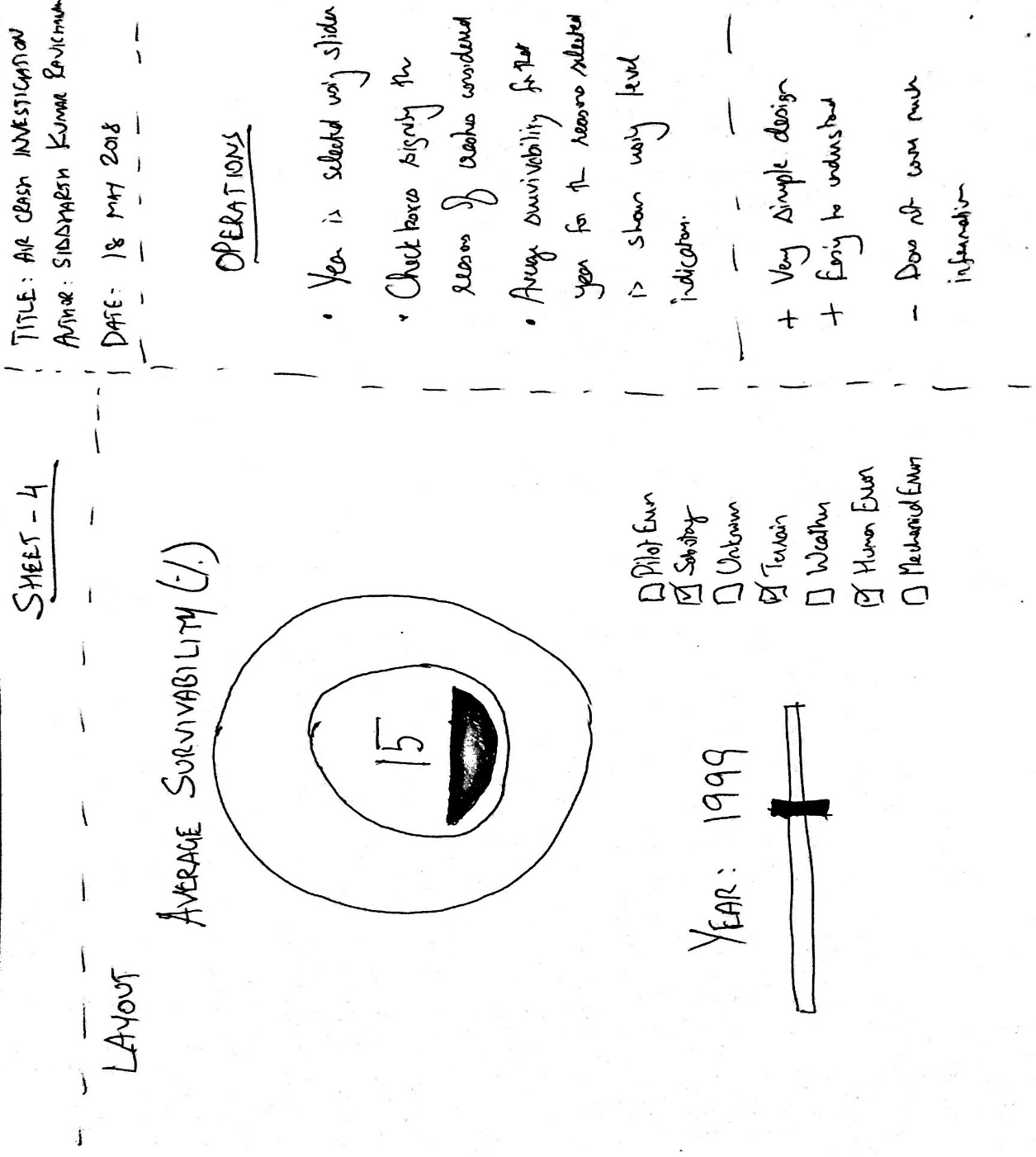
The user also could have an option to navigate through the years using a slider. Checkboxes could also be added to filter out the different crashes.

On hovering each crash, there could be a tooltip giving insight on the crash.



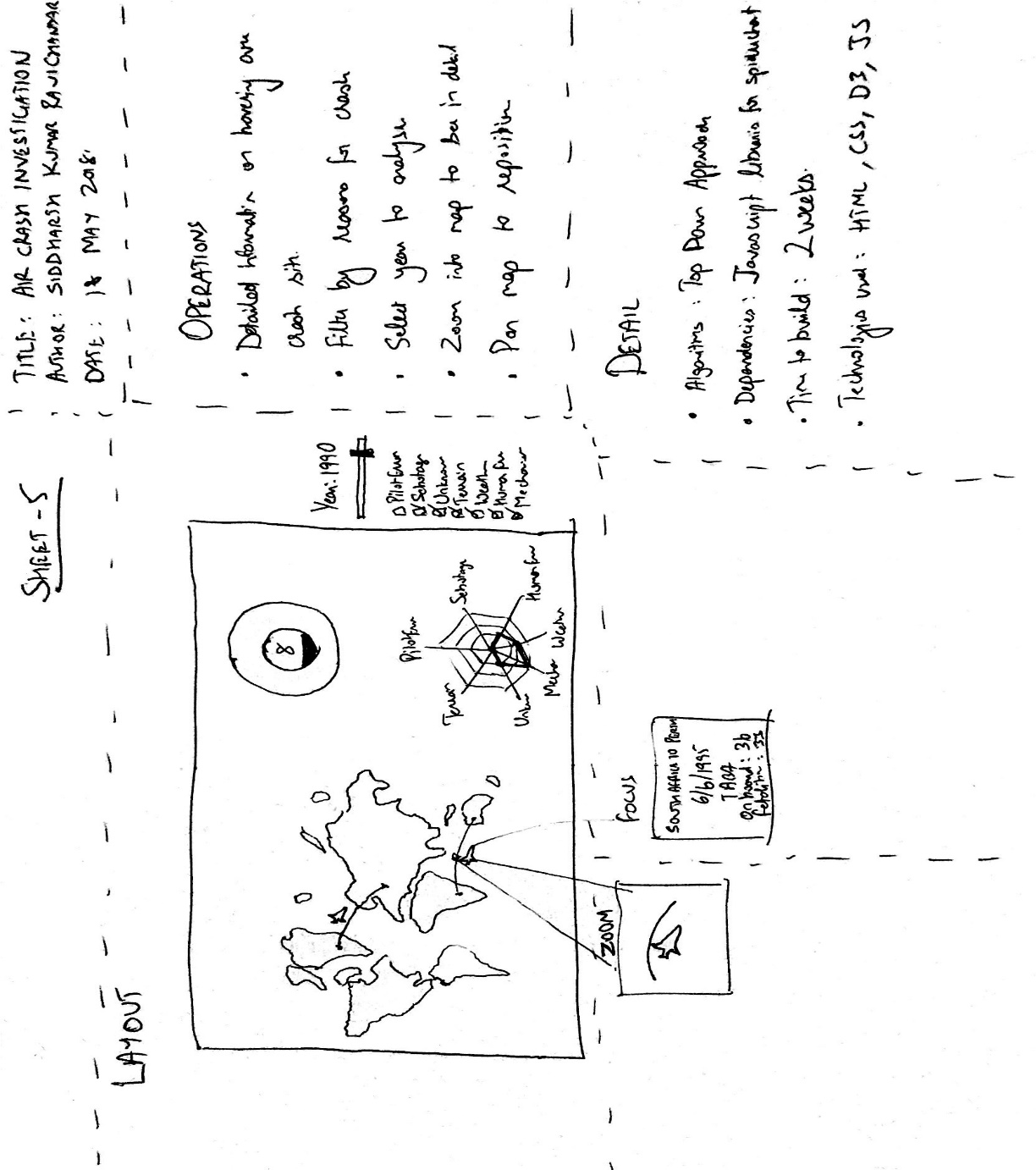
My second idea was to have a spider chart to show how each reason contributed to the crashes. This visualisation is very intuitive and easy to use for a normal person.

The user also could have an option to navigate through the years using a slider. Checkboxes could also be added to filter out the different crashes.



My third idea was to have a filling gauge to show how survivability varied by changing the year and the crash reason. This visualisation is very simple and easy to understand. But on the other side, it does not cover much information.

The user also could have an option to navigate through the years using a slider. Checkboxes could also be added to filter out the different crashes.



The designs from sheet 2, 3 and 4 were used to come up with a final design and operations. These 3 designs were chosen since it covers almost all the important data points and it is easy for the user to interact with.

It was decided to use the Top Down methodology for development.

A few Javascript scripts were used in the project, which is explained in detail in the implementation. It was predicted that it would take 2 weeks to implement the entire project.

It was decided to use HTML, CSS for web page building and styling. Javascript was used to make it user interactive. D3 was used to exploit its very appealing visualisations.

## Implementation

Obtaining geocodes for source and destination

Continuing from where we had left in the exploration project, I wanted to first plot a map with all the crash locations and also the routes they had taken. The crash locations were previously found while doing the exploration part of the project. But now, from the dataset, I had the route with location names, but I had to get the geocodes for the start point and intended destination of each flight that had crashed to plot the route on the map. I tried using the Datascience Toolkit as I had done earlier, but to my dismay, this time their servers seemed to be down. Hence, I proceeded with an alternative API provided by MapQuest. MapQuest had a daily limit of 3000 requests per day, while I had over 5000 crashes for which the source and destination had to be searched for. This would result in over 10,000 searches. To tackle this problem, I kept adding the search result to a dataframe. This dataframe would be checked before calling the API. In doing so, I was able to avoid calling the API for locations that I had already searched for.

D3

I had chosen to do the entire visualisation on D3. I felt this would be challenging and also very useful in my career in the future.

Plotting world-map on D3

I had chosen to do my visualisation on D3. So now, to plot the world map on SVG, I had to do the following steps:

1. Download shapefile - I was able to download a shapefile of the world map from the Natural Earth website : <https://www.naturalearthdata.com/downloads/110m-cultural-vectors/110m-admin-0-countries/>
2. Use GDAL to convert shapefile to GeoJSON
3. Convert GeoJSON to topojson

This topojson could then be used to plot on D3.

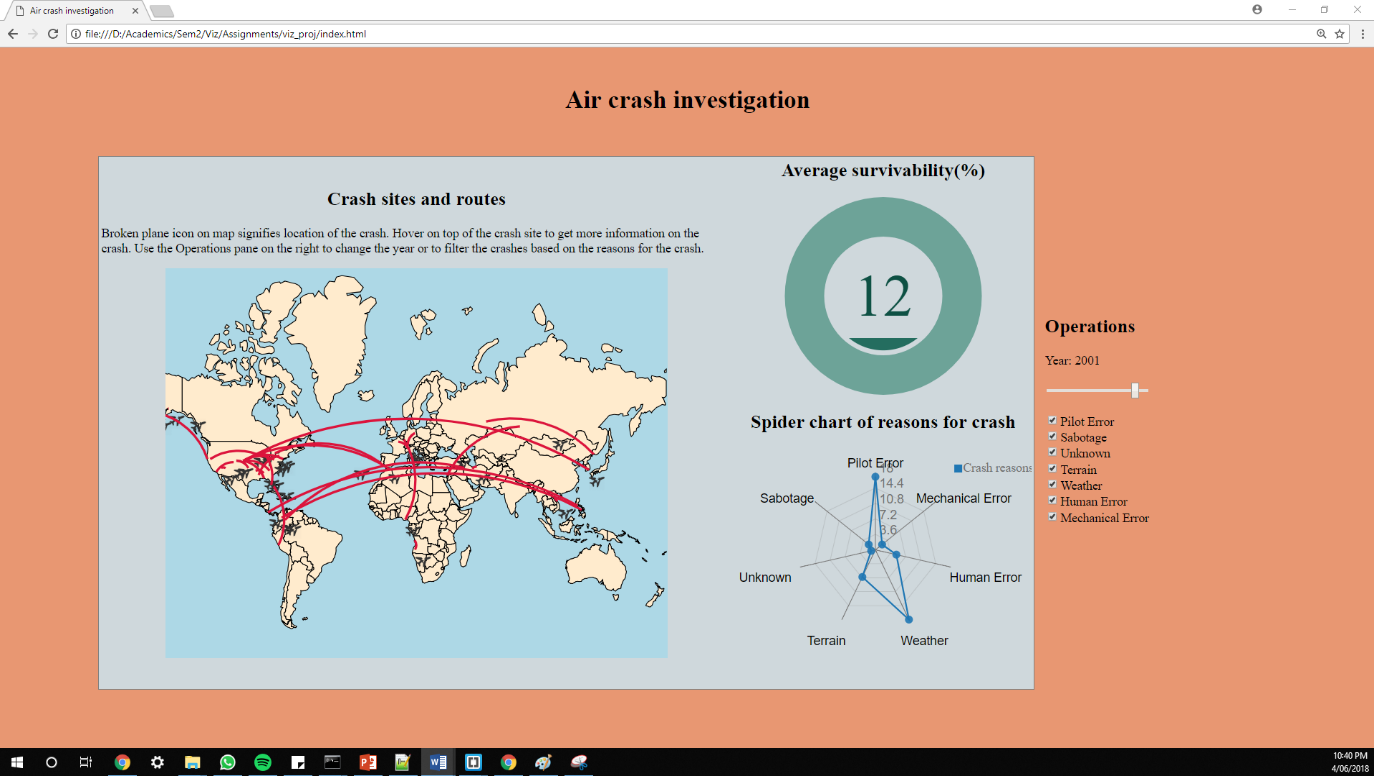
Spiderchart

A pre-written Javascript code (<http://bl.ocks.org/nbremer/6506614>) was used to visualise the spiderchart. This is a very well written script that has been progressively editted by multiple people. Writing this script from scratch would not be feasible given the short amount of time. Hence the script was used.

D3 Liquid Fill Gauge

A pre-written Javascript code (<http://bl.ocks.org/brattonc/5e5ce9beee483220e2f6>) was used to visualise the Liquid Fill Gauge. As mentioned earlier, writing this script from scratch would not be feasible given the short amount of time. Hence the script was used.

The final visualisation after implementation looks like below:



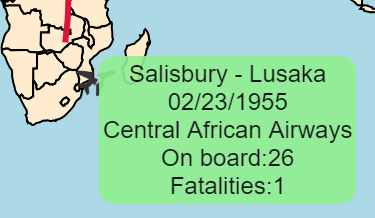
## User Guide



The red arc in the visualisation shows the path that each flight should have taken. In the above example, there are 5 paths and 5 planes. This means that we are looking at data of 5 crashes.

The location of crash is shown using the symbol:. One of the flights (Southern part of Africa) is circled in blue just to show how icon looks on the map.

On hovering the icon denoting location of crash, we can see a tooltip with more information on the crash. The tooltip is as shown below:



Upon hovering on a crash site, it only displays the crash site, flight path and tooltip of that particular crash. All other crash sites and flight paths are hidden. As compared to the first screenshot (which had 4 flights), there is only crash site and flight path of the crash hovered upon.

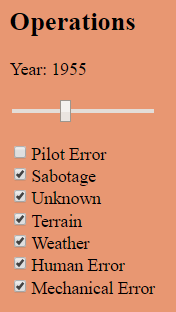


Upon moving the mouse away, the remaining crash sites and flight paths are shown again.

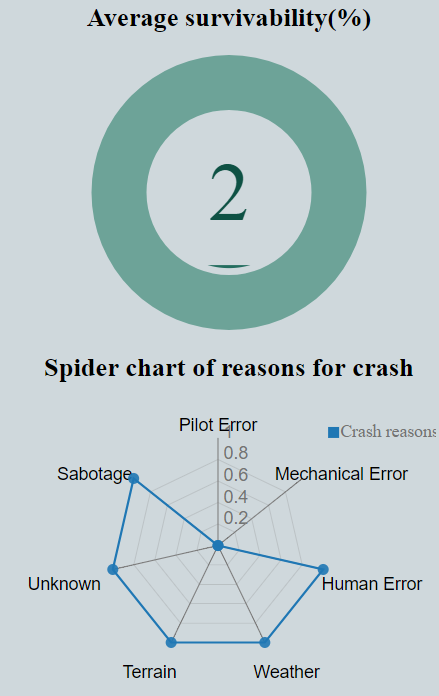
There is also a feature to pan (by clicking on map and dragging) and zoom (double click or scroll) into the map.

In this visualisation, although there is an arc showing the path that was to be taken by the flight, this is not the exact path. It is only made to look like a path. This is the reason why sometimes you will see that the crash location is nowhere near the path that the flight was supposed to be taking. Another reason why this happens is because the geocodes of the crash location was derived using the location names. This means sometimes the crash location might say Mediterranean Sea. But Mediterranean Sea is huge and covers multiple geocodes. But the API gives only one code for a location.

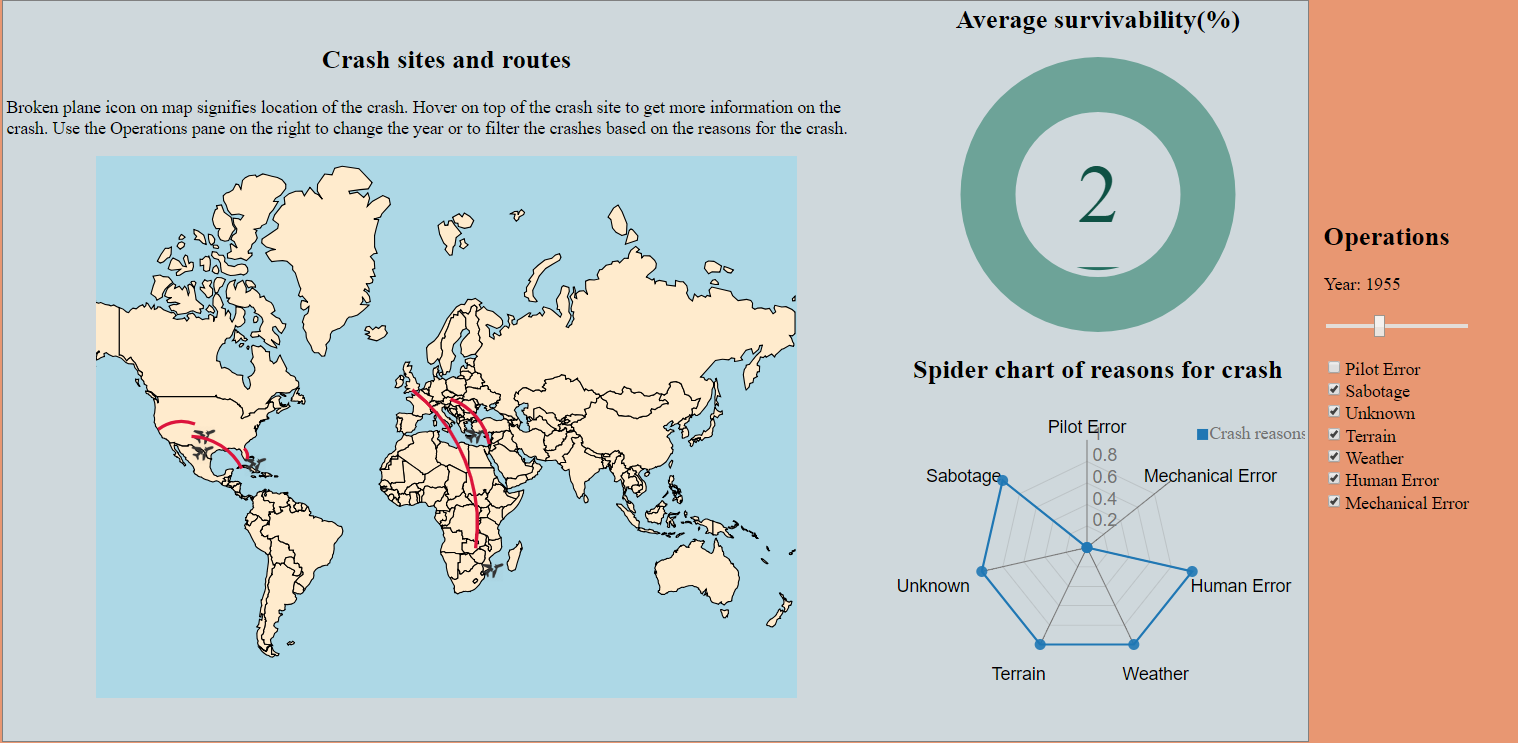
The operations pane on the right gives the flexibility to the user to change the years (using the range slider). It also allows users to filter out the crashes based on the reason for the crash using the check boxes. Shown below is a screenshot of the Operations pane:

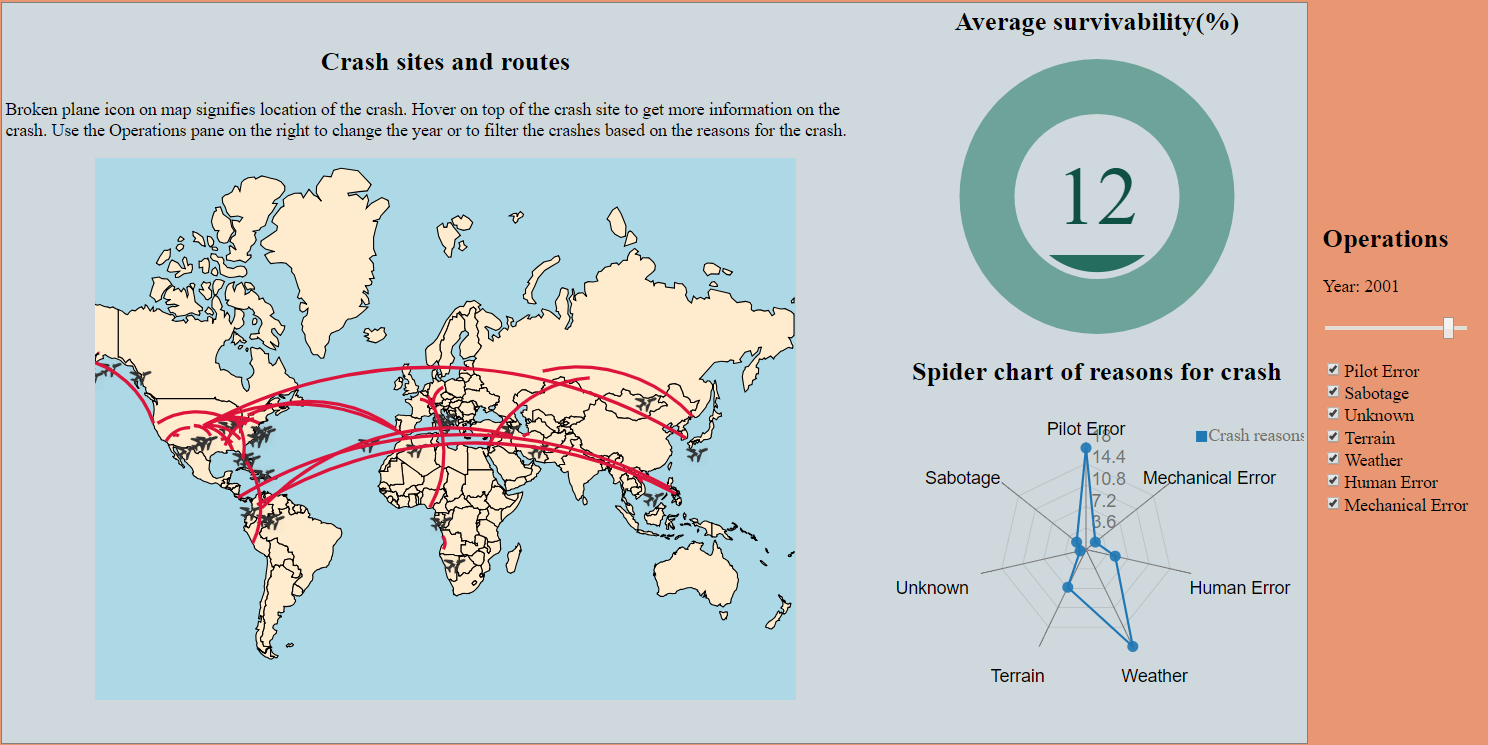


We also have visualisations to analyse the average survivability and a spider chart to see the frequencies of each of the reason of crash.



The operations mentioned earlier not only affects the map, but also of the Liquid Fill gauge and the spider chart as well. We can very clearly see the differences before and after changing the filter and the year on each of the visualisation as shown below:





## Conclusion

From the above 2 screenshots that were captured, we can easily see that the overall average survivability has increased multifolds over 50 years. We can also see that nowadays, most of the reasons for crashes seem to be either due to Weather or due to a fault with the pilot. This goes on to tell that most of the other factors for crashes has significantly reduced after taking necessary measures.

Overall, this project has made me learn a lot of new ways of finding interesting information from what seemed like not-so interesting data. I have also been exposed to very powerful ways of expressing information what you otherwise cannot express in words.

In hindsight, apart from what I have learnt and done in this project, given a chance, I would want to explore and find out how much has the air traffic increased over the years, budget allocation by each airline in maintaining existing flights and the general lifespan of flights.

## Why was the project difficult?

1. The dataset had textual data for comments on the crash. This had to be classified into various reasons by using a Support Vector Machine to train the data and then use the model on the test.
2. The process of converting location names to geocodes was a very tedious process. Since this had to be done for multiple columns – crash site, source of flight, intended destination.
3. The entire visualisation project was done on D3 which is very difficult when compared to RShiny.
4. User Interface was to be implemented on HTML, CSS, Javascript with no prior UI experience.
5. The visualisations proposed during the presentation required much more time than what was remaining.

## References and Bibliography

<https://medium.com/@ThomasG77/natural-earth-data-as-geojson-d15de47b8be1>

<http://bl.ocks.org/nbremer/6506614>

<http://bl.ocks.org/brattonc/5e5ce9beee483220e2f6>