CE201 Team Product Design, Implementation, and Testing Report

Team: Group 16

Team Members:

- **Nelvin Joseph**
- **Christopher Gray**
- **Matthew Noble**
- **Luis Wilkes Rodrigues**
- **Tanya Raju Ramchandani**

Project Manager: Luis Wilkes Rodrigues

Technical Lead: Nelvin Joseph

Team Supervisor (Academic): Professor Sam Steel

Table of Contents

Chapter 1 The Executive Summary (Christopher Gray, Nelvin Jospeh)

The following report contains details about how we worked on the completion of the project requested by the Education Committee (EC) of the CSEE department. The team have worked closely with an academic from the CSEE department to create a product to fulfil the needs of the EC. This product helps analyse the marks of students and help showcase what modules are hard and which ones are easy for the strong and weak students and as a whole.

The Project Manager section focuses mainly on the differences from the original Gantt chart against the reviewed one and elaborates on what caused the inaccuracy. In the Context section it goes into detail on how the product will be applied into real world scenarios and what impact our product may have for University Of Essex (UoE). The Conclusion expresses what us as a team have learnt from the project and whether we feel satisfied in meeting the EC request.

Chapter 2 Team Working

2.I The Team Activity Report (Nelvin Joseph, Luis Wilkes Rodrigues)

The team worked well together with the exception of Nikolas Tilemachou. Everyone equally contributed in team meetings and at the end of each team meetings we all decided on what tasks had to be completed by each person before the next meeting. Everyone had a chance to be the minute taker, and I was responsible for setting the agenda for each meeting. When it came to programming, we all split the work equally with Tanya Ramchandani working on import and bar charts, Matthew Noble to work on GUI, Christopher Gray on ranking the modules in difficulty, Luis Rodrigues to export the information to pdf and I was responsible for the pie chart and sorting the data for everyone to use. Also, all the team members contributed on other parts of the codes to tweak them and make them more readable and functional. Outside of meetings, we kept contact using WhatsApp.

The reason why there was an exception with Nikolas Tilemachou, was due to his consistence absence of work, effort and attendance towards the team. Despite there being this issue in the early stages, Nicolas only once contacted our team just to inform us that he was ill and therefore was unable to attend the early stages. Despite this comment the next week until now the end of the product there was no input or contact from him, even after our efforts. In result of this our team supervisor suggested our team to not include Nikolas within our team or product.

2.I.a The team effort summary table (see Appendix 1+2)

Chapter 3 Product Requirements Specification

3.I The Team Product SRS update (Luis Wilkes Rodrigues, Christopher Gray)

In comparison to the Product Specification the core ideas of outputting the analysis of student marks within different types of graphs and exporting the results in an appropriate form is correct. Despite following the same idea, there were only a couple of changes as a whole for the outcome specified in the Specification within the actual working product itself.

The differences were not forming the student analysis data in a line graph, as this was only done for bar charts and a pie chart. The reason why a line graph was not included was because it would not show any variation to the message and information portrayed within either the pie chart or the bar graphs. As well as this to make up for it, more bar charts were added to cover the full extent of the data. Additionally another difference was with there being two options to export the data, when in the actual product only pdf export was used, as csv wasn't an actual requirement. So therefore by only completing the pdf export, more time was then able to be spent on other areas within the project.

3.II References

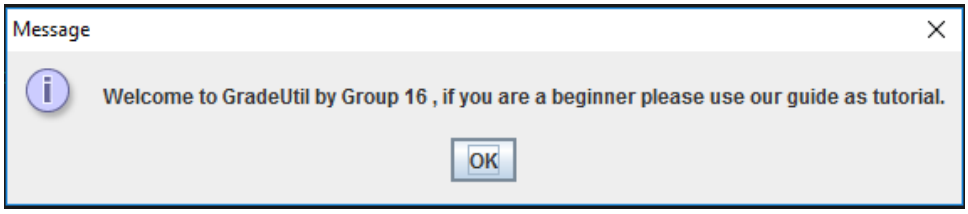
[1] The University of York. “IEEE Referencing Style”, york.ac.uk. [online] Available: **http://www.york.ac.uk/integrity/ieee.html** [Accessed: June 13, 2014]
[2] IEEE Recommended Practice for Software Requirements Specifications, IEEE Standard 830, 1998.

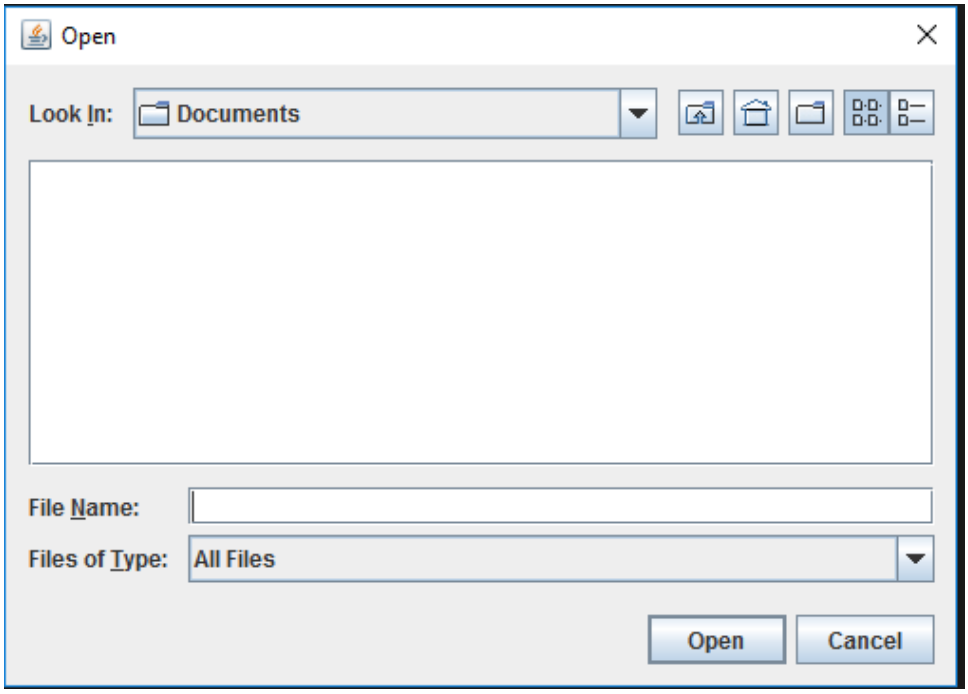
Chapter 4 Product Development (Tanya Raju Ramchandani)

4.I Design

The main consideration to get a good design solution was to have a good perspective in the users' eyes. So, good human interaction with the product was needed and graphic to make it more user friendly. The other thing that was considered was to satisfy the original functionality that was needed to satisfy the user/customer.

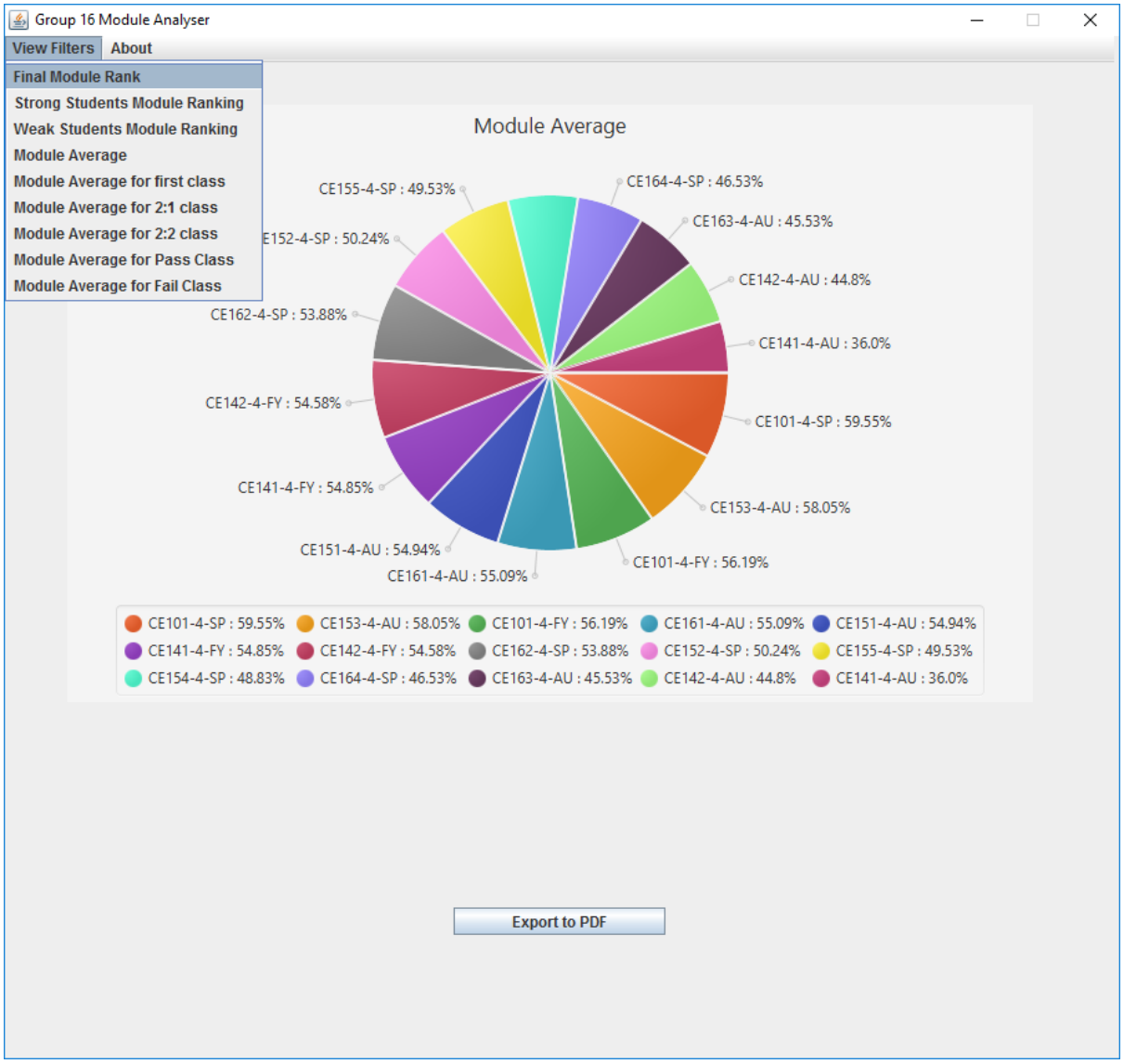
The program starts with greeting the user, and then asking for the file name that they would like to input/analyse, which was specified in the SRS report. And to select the file, user would need to double click the file or click it once and then click open.





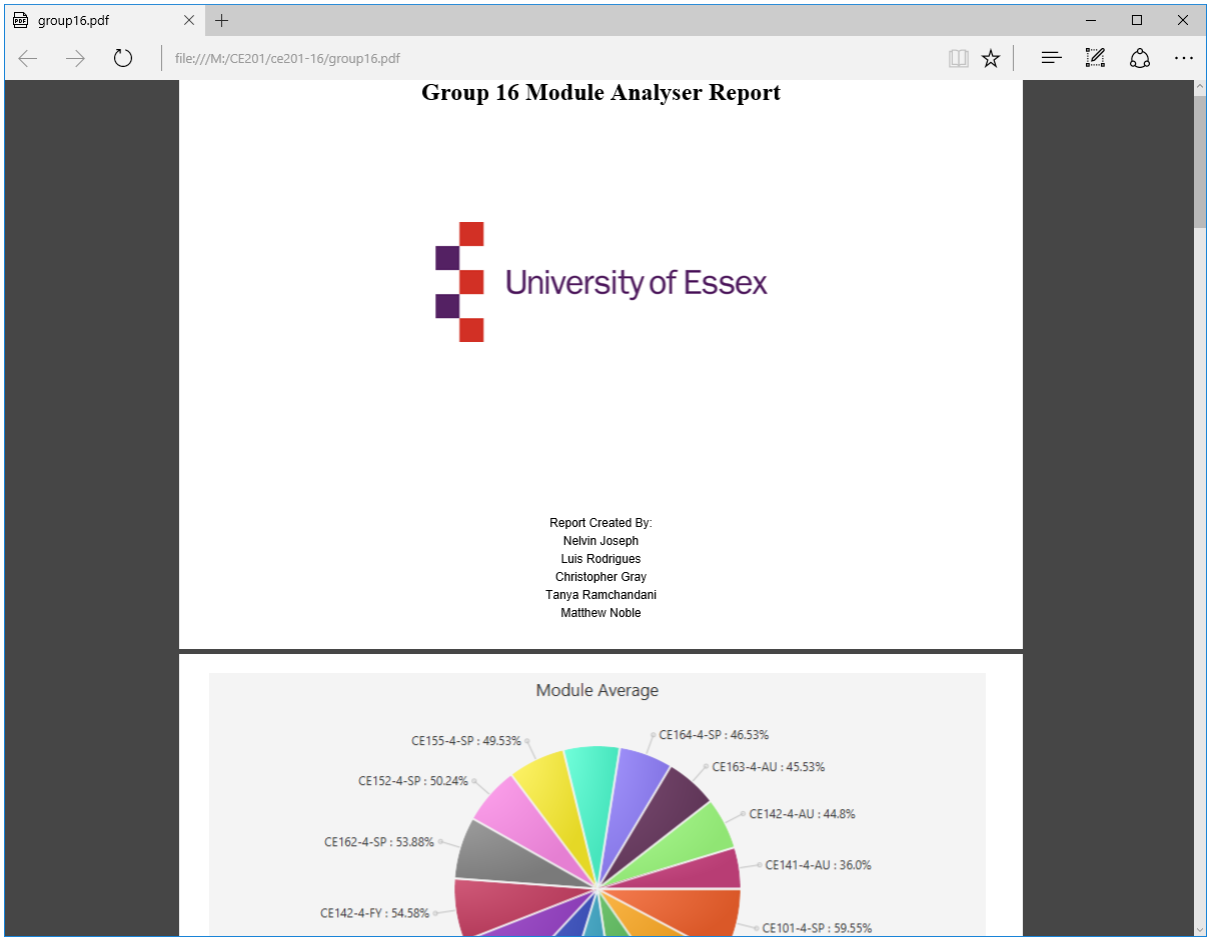
After the file is chosen, the file would be read virtually and then, the first thing that would be shown is a graph, with two menu bars in the top left side and an export button in the bottom. One menu bar has the option to select the graphs or data to see and the other menu bar has the guide to how to use the program. The colours of the pie chart were chosen according to what would be more user

friendly and satisfy the user.



And, instead of have the description of the graphs and data on the window, it is showed the pdf file after exported. It can be exported after all the tables have been seen for the screenshots to be taken. This is to keep it clean, tidy and short and more graphical. And, the pdf file would be saved in

the folder where the program is opened from. This is what the pdf file would look like.



4.II Implementation

Different parts of the code was done individually by the team members first, and then combined together in the end. This way was found to be very efficient and easier for all the team members. For importing the csv file that is chosen by the user, the file was read by a Buffer Reader line by line, and then added to an array list to read and use the data easily (Import.java). Then, to get the data as wanted(parsing), Tree maps were used to store the data for different charts that were to be made(Modules.java).

Another class for parsing data for creating the tables had comparators to have duplicate value in descending order(Weight.java). The tables needed were made by refer to Weight.java to have information in the order needed. The tables in Data.java gave the information of the hard and easy modules in general and for strong and weak students. So, it had three tables to show, with the hardest modules to be shown on top. The strong students were categorised as students who got first or 2:1, while the weak student with anything below that. These tables were implemented by getting the modules average marks of student who got 1st, 2:1, 2:2, etc. individually, which is also shown as graphs that have been implemented. And, the tables were also implemented by setting points to the modules, for example, first for that module would get least points and fail for the module would get maximum points. This means that the module with highest points would be the hardest module as most people must have failed or gotten low marks. Screenshot of Weight.java below:

```
        <~>(Collections.reverseOrder());
    public static TreeMap<String, Integer> weakWeight = new TreeMap<~>();
    public static TreeMap<String, Integer> finalWeakWeight = new TreeMap
            <~>(Collections.reverseOrder());

    public static void weighting() {
        int weigh = 1;
        int weighTO = 2;
        int weighTT = 3;
        int weighP = 4;
        int weighF = 15;

        for (Map.Entry<Double, String> entry : Modules.firstAverage.entrySet()) {
            String value = entry.getValue();
            firstWeighting.put(weigh, value);
            weigh++;
        }
        for (Map.Entry<Double, String> entry : Modules.secondAverage.entrySet()) {
            String value = entry.getValue();
            twoOneWeighting.put(weighTO, value);
            weighTO++;
        }
        for (Map.Entry<Double, String> entry : Modules.secondTwoAverage.entrySet()) {
            String value = entry.getValue();
            twoTwoWeighting.put(weighTT, value);
            weighTT++;
        }
        if (failWeighting.containsValue(name)){
            List<Integer> keys  = getKeysFromValue(failWeighting, name);
            weight += keys.get(0);
        }
        weakWeight.put(name, weight);
        weight = 0;
    }

    Comparator<String> valueComparator =  new Comparator<String>() {
        public int compare(String k1, String k2) {
            int compare = finalW.get(k2).compareTo(finalW.get(k1));
            if (compare == 0) return 1;
            else return compare;
        }
    };
    finalWeighting = new TreeMap<String, Integer>(valueComparator);
    finalWeighting.putAll(finalW);

    Comparator<String> valueComparator1 =  new Comparator<String>() {
```

Then, we have all the bar charts showing the module averages of students with 1st, 2:1, 2:2, 3rd and failed individually. This concludes having 5 bar charts of the information as said above. Also, one pie chart is included to the module average as a whole to get a general idea of the student in that year that has been selected by the user. The charts (Bargraph.java-Bargraph5.java and pieCharts.java) were created using JavaFX and getting the parsed data from the Tree maps from Modules.java. Also, to have different colours for the pie chart slices and bar charts, a CSS file was used to change that. This was mainly done because the colours in the pie chart were repeating, which would make data difficult to distinguish, so it was decided that the colours in the bar chart must also be changed to avoid bright colours that are hard to looks at.

```
private static void runFX(JFXPanel fxPanel){
    Scene scene = create();
    scene.getStylesheets().add("chartColor.css");
    fxPanel.setScene(scene);
}

/*
*CREATES THE BAR CHART AND ADD EACH VALUES TO IT
*/
private static Scene create(){
    final CategoryAxis xA = new CategoryAxis();
    final NumberAxis yA = new NumberAxis();
    final BarChart<String, Number> barChart = new BarChart<~>(xA,yA);
    barChart.setTitle("Module Average for 2:1 class");
    barChart.setLegendVisible(false);
    xA.setLabel("Module Name");
    yA.setLabel("Module Average");
    barChart.setAnimated(false);

    XYChart.Series series = new XYChart.Series();
    series.setName("Modules");
    for (Map.Entry<Double, String> entry : Modules.secondAverage.entrySet()) {
        Double key = entry.getKey();
        String value = entry.getValue();
        if (Double.isNaN(key)) {
            key = 0.0;
        }
        series.getData().add(new XYChart.Data(value, key));
    }
    barChart.getData().add(series);
    Scene scene = new Scene(barChart,775,480);
    scene.getStylesheets().add("chartColor.css");
    ChartImg.saveBarAsPng(barChart, "chart2.png");
    return (scene);
```

For exporting the information as pdf file, screenshots of the graphs and table were needed. So, the methods were made for them in ChartImg.java. To add information to pdf, Export. Java class was created, adding everything and other words that were needed. Lastly the GUI.java class, the graphical part was created using panels that added the graphs and tables and a button the export the data as pdf. The GUI class had the file chooser input that would run all the methods needed and the main method Run.java would run the GUI.java class that runs the entire program.
The technical achievement that the team was proud of was learning and using JavaFX for creating the graphs. This was something new and all of understood them easily. The other thing that we learnt was to take screenshots of images from the panel itself and of the graphs.

4.III Testing

Test	Goals	Pass/Fail?	Solution/Improvements
Test whether CSV import and parsers work(Import.java, Module.java, Weight.java) Done by outputting data(array list and Tree	Output the information that was tried to input in the arraylist and Tree maps	Pass, therefore can be used in implementing the charts and tables	NA

maps) using System.out.println			
Test all the bar charts(all Bargraph.java classes) after the information has been collected using the above classes	Charts should give the information that is added to it	Fail (some fields were blank)	Solution: created if conditions so that it can ignore the data that with no students taking the module
Test the bar charts again	Charts should give the information that is added to it	Pass	NA
Test pie chart after the information has been collected using the above classes	Pie chart should have 15 different colours and give the information that is added	Fail	Solution: made a CSS file so that the colours in the pie chart wouldn't repeat but have new colours
Test the pieChart.java again	Pie chart should have 15 different colours	Pass	NA
Test tables(Data.java) after the information has been collected using parser classes in the first row	Tables should give the information that have been added to show	Fail (some information were not shown)	Solution: created if conditions so that it can ignore the data that has no entry as the students don't take by checking NaN
Test tables again	Tables should give the information that have been added to show	Pass	Solution: created new class with methods to take screenshots of the graphs and tables to add to pdf
Test the Export.java	Pdf file must be made with the charts, tables and words added	Fail (charts and tables were not shown)	Solution: created new class with methods to take screenshots of the graphs and tables to add to pdf
Test Export.java and ChartImg.java(screenshots class) together	Pdf file must be made with the charts, tables and words added	Fail (2 out of 3 tables have a black background)	Improvements for future: make a different method for screenshot of tables that avoid the black background
Test GUI after combing everything	All the buttons must be functional directing us to what	Pass	NA

	was chosen		
Test it as a jar file	Everything must work	Fail (due to the file path of the csv file)	Solution: made a file chooser, which was the original plan
Test it as a jar file again	Everything must work	Pass	NA
Test the file choose with file format other than csv	Shouldn't work with other file formats	Fail	Improvements for future: have restriction on file type in file chooser

All this testing was done after all the team members had done their part individually.

Chapter 5: Project Management (Luis Wilkes-Rodrigues)

5.I Project Management Report

5.I.a The team Gantt charts (See Appendix 3)

5.I.b A discussion of the Gantt charts - (Luis Wilkes-Rodrigues)

In comparison between the first Gantt chart and the second Gantt chart it is clear that the initial plans were optimistic in the timing of work scheduling. There would've been a few explanations to this for example as it was early in the stages of the group, not everyone knew each other's strengths well, didn't plan for any major absences. Despite changes in the time scheduling of work the improved upon Gantt chart only has a few more stages added to it and this is due to information about the project being presented in later stages of the project as well as being unable to predict when future team meetings would be scheduled.

For each of the Gantt charts I used milestones for start and end dates of each task along with milestones for each of their testing dates as they key parts in the project. Milestones where also used to highlight when a team meeting was scheduled as good team communication is key within a group project. Additionally the main content of the Gantt charts were all set up using auto-scheduling as well as setting a few dependant and independent tasks among the project. This is due to the nature of some of the tasks as information may have been required from a previously completed task within the timeline of the project.

5.II.c An evaluation of the project management - (Luis Wilkes-Rodrigues)

The design and build progressed in the order as expected to our Agile Methodology starting with planning, implementation and testing for the project. Despite following the order well, in some areas extra time was spent where in an ideal situation each set in the methodology could've been finished earlier. Although I believe that as a team we have succeed in a result of using the Agile Methodology due to testing being completed after each area in the project is completed.

During each of these stages I had to make sure as a Project Manager to check that the work set was fair throughout the team and that each member's contribution would be beneficial for the project's result. Each week I checked the minutes and agenda as well as the work progress from team members. Despite this being done I was unable to complete this work for every member in the team as one of our team members Nicolas was absent for a large majority of our labs and team meetings. Therefore it was a rare occasion in which I would seem him and there was no work contribution to be checked in those cases.

Chapter 6: Context (Matt)

6.I Legal

As a team, group 16 collectively coded individual elements of which made the program. It was mutually agreed and further more understood that in a professional reality the ownership of the code would be handed to the University Of Essex as we have essentially been approached by them to mimic a real world software development process. In terms of licencing the university would hold the right to apply the program when and how it pleases.

6.II Ethical

Collectively it is assumed that group 16 had acted in a whole fully ethical manner throughout the whole process. From the very beginning of the project, care was taken in making all voices heard. This was something that as a team we all fully upheld this principle, the team acted in a very intersectional and inclusive way, something I feel must be given as testament to the Team leader Nelvin Joseph. Group 16 did not exploit any individuals with regards to work allocation. This being said we did unfortunately have to severe ties with past team member Nicolas, as he was not acting conducive to the teams future. During the first term, to coerce Nicolas into the team in a friendly manner in both non and formal ways, with the latter being documented on the team forum. Through-out the whole academic year we have seen Nicolas once and so with the advice from Supervisor Sam Steel we unfortunately ended our ties with him. The focus and main ideal of group 16 was to uphold and endorse a strong, healthy team dynamic in the most inclusive way possible.

6.III Health and Safety

When looking at the case of health and safety in connection to group 16's product, after testing it is evident that the software's threat in regards to individual health is completely no-existent. To the user we could find zero possible triggers in regards to epilepsy of which could have potentially formed themselves with the use of vivid colours that would move quickly. In terms of computer health our program is not terribly strenuous to the computers functionality and so in reality has minimal risk to an individual's safety in even the most conservative of assumptions.

In the development of our product, the general health and safety recommendations given to the workplace by the UK government via the Health and Safety at Work act of 1974. Specifically it is advised that long periods of computer use should be countered by regular break intervals. [3]

[1]Data Protection Act 1998 http://www.legislation.gov.uk/ukpga/1998/29/contents

[2]7.8 IEEE Code of Ethics http://www.ieee.org/about/corporate/governance/p7-8.html

[3]Health and Safety at Work etc. Act 1974 http://www.legislation.gov.uk/ukpga/1974/37/contents

Chapter 7: Conclusions (Matt)

To conclude this report, it is essential to say that in our field, teamwork is at the base of any great accomplishments. Acting as a bond between our struggles to develop a good product that not only satisfies, but also exceeds the average user's expectations, teamwork was adequately interlaced with our hard work and dedication. Each member of the team played a well-defined role in the process of developing a marketable product.

The product development process that we went through as a team was divided into five stages, as imposed by the agile methodology: specification stage, design, testing, implementation, testing, and maintenance testing. While elaborating our application, we undertook various research tasks that helped us deliver a quality product by the deadline. Our area of research involved practical solutions for implementation problems as well as general principles for good software development. As it can be observed, there was a major change in our way of approaching the product development. But by the end of the project, the whole product came together, showing that our cumulated efforts were well worth it.

As we have realised so far, the managerial part of a project holds a very important role. In achieving such a complex task, we found ourselves in need for efficient collaboration, and as the collaboration in a specialised team does not come naturally, we ended up relaying on our project manager's sense of organisation even more than we would have expected. He helped us become a team, and this way we could succeed in finishing the project. Another discovered importance of how planning and organising could make process more efficient was the dramatic improvement and progress that happened after we have scheduled out workload. To-do lists and prioritisation helped us complete the product on time. Other organisational mechanisms were implemented, as an updated logbook that kept track of our actions, and a Gantt that better systematise our schedule. In the end, we can all have a better appreciation for the importance of our manager, as he helped us achieve a certain degree of organisation and target our efforts the right way.

Appendix 1: The Team Effort Summary Table

Member		Nelvin Joseph	Luis Rodrigues	Tanya Ramchandani	Christopher Gray	Matthew Noble
Role		Team Leader	Project Manager	Specialist	Specialist	Specialist
Chapter	**Sub-section**					
Executive summary		**R**	**C**	**C**	**R**	**C**
Team Working	Team activity Report	**R**	**R**	**I**	**C**	
SRS		**C**	**R**		**R**	
Product Development	Design	**C**	**C**	**R**		
	Implementation	**C**	**C**	**R**		
	Testing	**C**	**C**	**R**		
Project Management	Report	**C**	**R**			
	Evaluation	**C**	**R**			
	Gantt Charts	**C**	**R**			
Context	Legal	**C**	**C**			**R**
	Ethical	**C**	**C**			**R**
	H+S	**C**	**C**			**R**
Conclusions		**C**	**C**	**C**	**C**	**R**

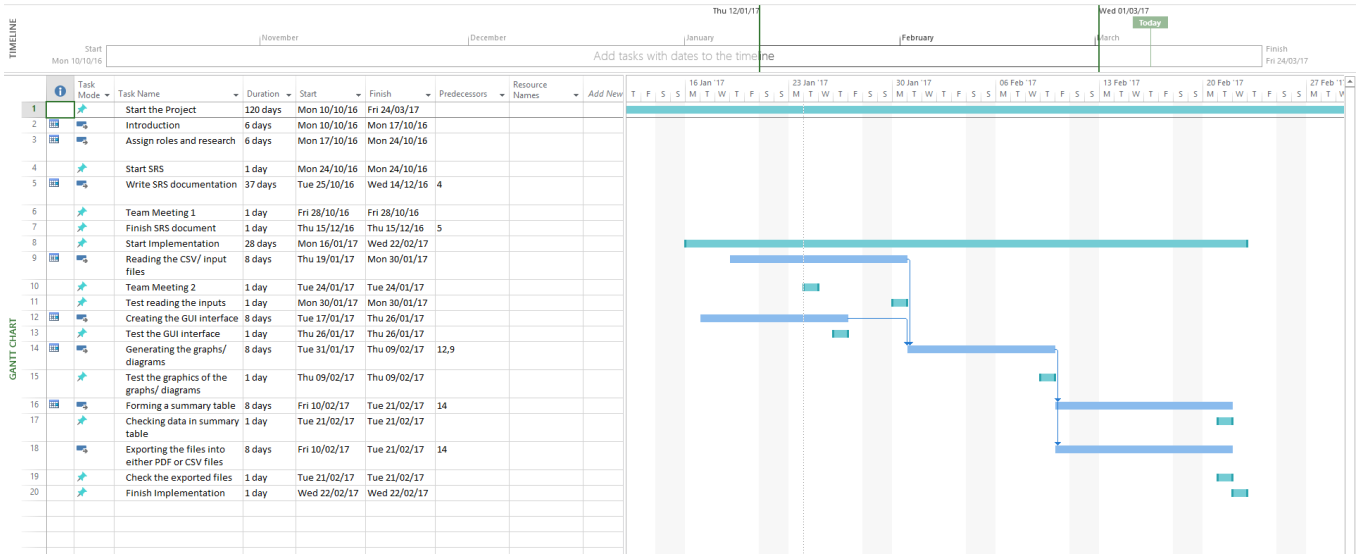
R = Responsible
C= Consulted
I = Involved

Appendix 2: The Coding Effort Summary Table

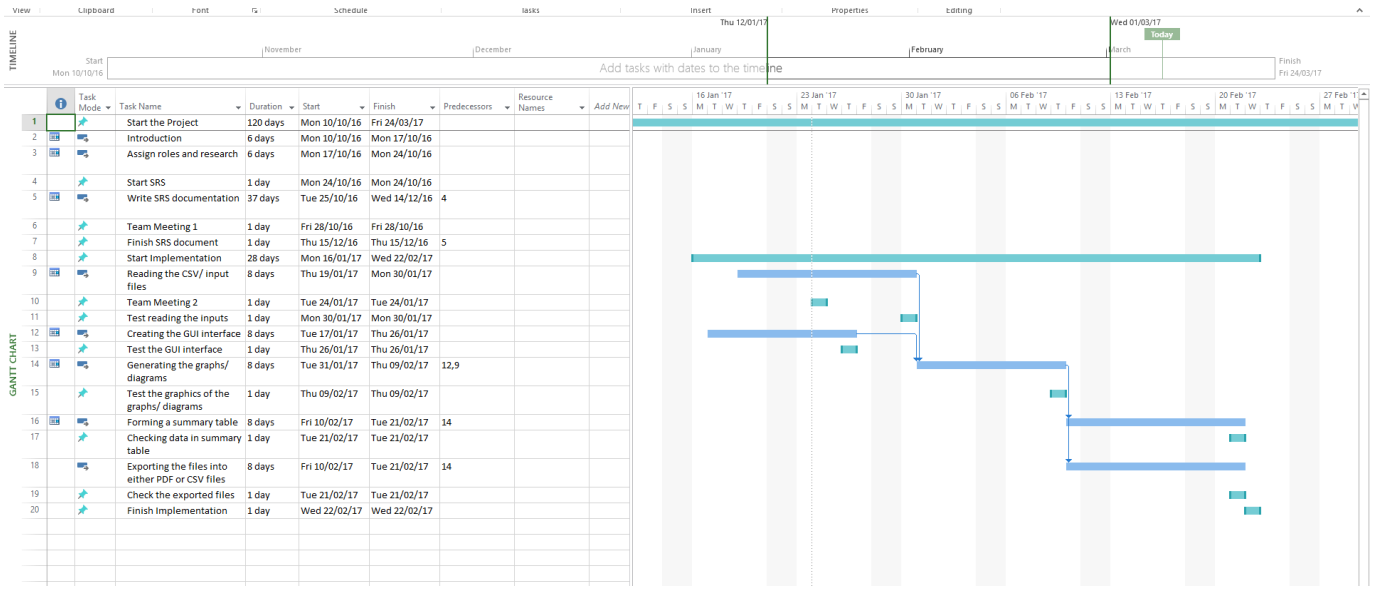
Member	Nelvin Joseph	Luis Rodrigues	Tanya Ramchandani	Christopher Gray	Matthew Noble
Role	Team Leader	Project Manager	Specialist	Specialist	Specialist
Java Class					
Import	**I**	**C**	**R**		
Modules	**R**	**C**	**C**		
PieCharts	**R**	**I**		**I**	**C**
BarGraph	**R**	**I**	**R**	**C**	**C**
BarGraph2	**C**	**I**	**R**	**C**	**C**
BarGraph3	**C**	**I**	**R**	**C**	**C**
BarGraph4	**C**	**I**	**R**	**C**	**C**
BarGraph5	**C**	**I**	**R**	**C**	**C**
Weight	**I**	**C**	**I**	**R**	
Data	**I**	**I**		**R**	
GUI	**I**	**C**	**I**		**R**
Run	**I**	**I**	**I**	**I**	**I**
ChartImg	**C**	**R**	**R**		
Export	**C**	**R**	**C**	**I**	

R = Responsible
C= Consulted
I = Involved

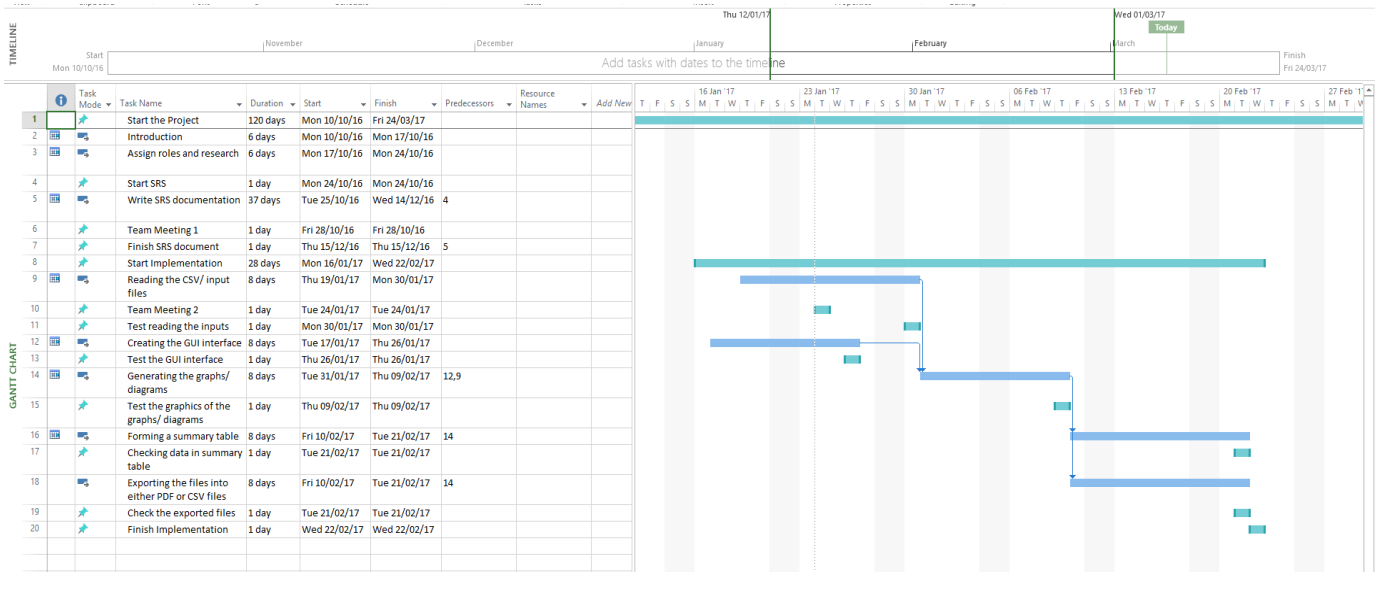
Appendix 3: The Team Gantt Charts (Luis Wilkes Rodrigues)



	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		Start the Project	120 days	Mon 10/10/16	Fri 24/03/17		
2		Introduction	6 days	Mon 10/10/16	Mon 17/10/16		
3		Assign roles and research	6 days	Mon 17/10/16	Mon 24/10/16		
4		Start SRS	1 day	Mon 24/10/16	Mon 24/10/16		
5		Write SRS documentation	37 days	Tue 25/10/16	Wed 14/12/16	4	
6		Team Meeting 1	1 day	Fri 28/10/16	Fri 28/10/16		
7		Finish SRS document	1 day	Thu 15/12/16	Thu 15/12/16	5	
8		Start Implementation	28 days	Mon 16/01/17	Wed 22/02/17		
9		Reading the CSV/ input files	8 days	Thu 19/01/17	Mon 30/01/17		
10		Team Meeting 2	1 day	Tue 24/01/17	Tue 24/01/17		
11		Test reading the inputs	1 day	Mon 30/01/17	Mon 30/01/17		
12		Creating the GUI interface	8 days	Tue 17/01/17	Thu 26/01/17		
13		Test the GUI interface	1 day	Thu 26/01/17	Thu 26/01/17		
14		Generating the graphs/ diagrams	8 days	Tue 31/01/17	Thu 09/02/17	12,9	
15		Test the graphics of the graphs/ diagrams	1 day	Thu 09/02/17	Thu 09/02/17		
16		Forming a summary table	8 days	Fri 10/02/17	Tue 21/02/17	14	
17		Checking data in summary table	1 day	Tue 21/02/17	Tue 21/02/17		
18		Exporting the files into either PDF or CSV files	8 days	Fri 10/02/17	Tue 21/02/17	14	
19		Check the exported files	1 day	Tue 21/02/17	Tue 21/02/17		
20		Finish Implementation	1 day	Wed 22/02/17	Wed 22/02/17		



	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		Start the Project	120 days	Mon 10/10/16	Fri 24/03/17		
2		Introduction	6 days	Mon 10/10/16	Mon 17/10/16		
3		Assign roles and research	6 days	Mon 17/10/16	Mon 24/10/16		
4		Start SRS	1 day	Mon 24/10/16	Mon 24/10/16		
5		Write SRS documentation	37 days	Tue 25/10/16	Wed 14/12/16	4	
6		Team Meeting 1	1 day	Fri 28/10/16	Fri 28/10/16		
7		Finish SRS document	1 day	Thu 15/12/16	Thu 15/12/16	5	
8		Start Implementation	28 days	Mon 16/01/17	Wed 22/02/17		
9		Reading the CSV/ input files	8 days	Thu 19/01/17	Mon 30/01/17		
10		Team Meeting 2	1 day	Tue 24/01/17	Tue 24/01/17		
11		Test reading the inputs	1 day	Mon 30/01/17	Mon 30/01/17		
12		Creating the GUI interface	8 days	Tue 17/01/17	Thu 26/01/17		
13		Test the GUI interface	1 day	Thu 26/01/17	Thu 26/01/17		
14		Generating the graphs/ diagrams	8 days	Tue 31/01/17	Thu 09/02/17	12,9	
15		Test the graphics of the graphs/ diagrams	1 day	Thu 09/02/17	Thu 09/02/17		
16		Forming a summary table	8 days	Fri 10/02/17	Tue 21/02/17	14	
17		Checking data in summary table	1 day	Tue 21/02/17	Tue 21/02/17		
18		Exporting the files into either PDF or CSV files	8 days	Fri 10/02/17	Tue 21/02/17	14	
19		Check the exported files	1 day	Tue 21/02/17	Tue 21/02/17		
20		Finish Implementation	1 day	Wed 22/02/17	Wed 22/02/17		



	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		Start the Project	120 days	Mon 10/10/16	Fri 24/03/17		
2		Introduction	6 days	Mon 10/10/16	Mon 17/10/16		
3		Assign roles and research	6 days	Mon 17/10/16	Mon 24/10/16		
4		Start SRS	1 day	Mon 24/10/16	Mon 24/10/16		
5		Write SRS documentation	37 days	Tue 25/10/16	Wed 14/12/16	4	
6		Team Meeting 1	1 day	Fri 28/10/16	Fri 28/10/16		
7		Finish SRS document	1 day	Thu 15/12/16	Thu 15/12/16	5	
8		Start Implementation	28 days	Mon 16/01/17	Wed 22/02/17		
9		Reading the CSV/ input files	8 days	Thu 19/01/17	Mon 30/01/17		
10		Team Meeting 2	1 day	Tue 24/01/17	Tue 24/01/17		
11		Test reading the inputs	1 day	Mon 30/01/17	Mon 30/01/17		
12		Creating the GUI interface	8 days	Tue 17/01/17	Thu 26/01/17		
13		Test the GUI interface	1 day	Thu 26/01/17	Thu 26/01/17		
14		Generating the graphs/ diagrams	8 days	Tue 31/01/17	Thu 09/02/17	12,9	
15		Test the graphics of the graphs/ diagrams	1 day	Thu 09/02/17	Thu 09/02/17		
16		Forming a summary table	8 days	Fri 10/02/17	Tue 21/02/17	14	
17		Checking data in summary table	1 day	Tue 21/02/17	Tue 21/02/17		
18		Exporting the files into either PDF or CSV files	8 days	Fri 10/02/17	Tue 21/02/17	14	
19		Check the exported files	1 day	Tue 21/02/17	Tue 21/02/17		
20		Finish Implementation	1 day	Wed 22/02/17	Wed 22/02/17		