CE201 Team Product Design, Implementation, and Testing Report

Team: 1C

Team Members:

- **Dale Carr**
- **Evaldas Senavaitis**
- **Benjamin Higgs**
- **Joshua Bloom**
- **Samuel Carslake**
- **Joshua Oluwayemi-Coker**

Project Manager: Dale Carr

Technical Lead: Evaldas Senavaitis

Table of Contents

Chapter 1 The Executive Summary (353 words)

This document is the team report on Design, implementation and testing. It contains all the relevant information regarding these subjects, and also details on team management.

Chapter two of this document will detail the work of the team members and their contributions to the group. It lists the details of everyone's commitments to the group project, of which there should be evidence of throughout this report and on the Moodle.

Chapter three is reporting of the changes to our original SRS documents. There are a number of changes that have been made since the original SRS was submitted. These changes are mainly based around adding another export to the system. We decided to add a TXT output too for a future, partially implemented use case.

Chapter four of this report contains the details regarding design, implementation and testing. These, on a whole, were very successful. The design of the system is that of a good, modular and professional structure. The design plans include various diagrams that relate in both low and high level design. The class diagram is an example of high level design. It shows the planned class structure, inner class listeners, variables and methods. The software implementation is successful and works well. The program matches all requirements and exceeds many too. The implementation matches the design that we set out for the project. We chose a specific selection of libraries to use for the system, which enabled us to create the systems key features. The testing part shows that the system was a success, and how it was bug free. This demonstrates that the test driven development methodology used by the programmers was a success. The tests allow us to ratify the working of the system, and prove that all works as planned.

Chapter five will go into detail about the project management, highlighting how well the product was implemented according to an early plan, as well showing the differences and similarities between our preliminary Gantt chart and the one created during development. The conclusion will summarise the development of the product and how the team felt about the end result.

Chapter 2 Team Working (1100 words)

2.I The Team Activity Report

Benjamin Higgs

Benjamin Higgs was assigned the role of programmer. Over the course of the module, he did the majority of the programming, and either built, or collaborated with the other programmers, to build all elements of the program. He played a large part of the systems design, creation, testing and documentation.

When the team worked on the SRS document, he created the perspective section, which contained detailed plans and research regarding the specific interfaces the program will work with. This section of the report aided the programmers. He also wrote two more sections. One based on the program

external interfaces, which detailed information about the inputs and outputs of the program. The second was the software attributes which went into detail about modularity, data handling, reliability, and data types. For the final report, Benjamin completed the design section. This was completed with all relevant diagrams related to the systems class structure, states, data flow and execution. Anything that was not diagrammed was explained using a high level design.

On the Moodle, Benjamin uploaded documents related to the data structure and flow plans for the final program. This helped the programmers to determine which data structures to use and how to store and iterate through complex objects.

Benjamin designed and programmed most of the system. His modular programming style allowed for efficient development, and enabled debugging of specific elements to occur. When it came to testing, he designed the test plans, as he had the biggest insight into how the program would work. This enabled Joshua to complete detailed testing into each sub section of the program, and attempt to break it.

Joshua Oluwayemi-Coker

Joshua was initially put under the category of one of the programmers, but as there were three other programmers who were more proficient and reliable, this resulted him in participating in other tasks of the projects such as the constraints, assumptions & dependencies, events, concurrency and control flow and the testing of the program. The main output from him came from the testing of the program which was created by Benjamin Higgs and the program was then undertaken and proven using screenshots by Joshua Oluwayemi-Coker. The testing helped show not only what the GUI excelled in but what it also lacked in, in order for them to improve this in the future.

Dale Carr

Dale was designated project manager of the team. Due to less coding experience than others in the group, he took a role allowing him to hold back on the coding front, but still have input within the team. Updating the agenda each week, as well as taking larger roles in the SRS and team report and the presentation, he has stayed up to date with the product development.

For the SRS, he wrote the user characteristics section as well as the overview. The user characteristics was to highlight the qualities that the customer would (or should) possess in order to accurately utilise the product. The Overview section was meant to break down the work in the other sections, giving a brief look at the document.

For the presentation, he co-created the document and delivered a portion of it before the customer, describing in detail the GUI as well as the future implementation that the group had thought of.

On Moodle, The Agenda has been updated weekly, ensuring the team was kept up to date with the current status of the product, as well as any other deliverables that were due during that time.

Samuel Carslake

Sam's role within our team was a developer thus one of the primary goals for him within this project was to help turn the ideas we had into an actual product. For the programming side of the project Sam did very little, this is due to him being rather bad at programming and making a conscious decision not to get involved in the programming side. Sam's decision behind this was that he felt as if he would bring down the quality of the overall product and thus make it inferior to others.

Since Sam didn't partake in much of the actual programming he did a few of the writing tasks on both the first deliverable (1.1 Purpose, 2.4 Constraints and 3.1 User Interfaces) his main contribution for the second deliverable has been Chapter 3 SRS Update. These can be shown on both the Deliverable 2 Team Effort Summary table and the Deliverable 1 Team Effort Summary table.

Sam was also assigned the role of Secretary and for some of the meetings he was chairperson

Evaldas Senavaitis

Evaldas Senavaitis was assigned the role of Technical lead. Over the course of the module, he collaborated with Benjamin Higgs to come up with extra features, details and fine tuning of the software leading to create fully working product. In a team he usually assigned tasks and planned next official meetings, allowing each member to choose which tasks they want to do in collaboration with "minute taker" for documentation. He as a technical lead had mayor impact how the software would come out for a consumer and played a large part of the system design as well as creating it.

When SRS document was created, he was responsible for the whole scope of the product and involving in creating product functions. As his role required him to overlook all other programmers, he had to be up to date to what other programmers where doing. Leading to implementation part of the software he and Benjamin did most of the software design and functionality, mostly working in official meetings.

He was responsible for GUI and any other parts that other programmers would be stuck on. While Benjamin created the prototype program other programmers were signed to work on that by Technical lead, as he saw that it was well done and it can be developed further on. He specifically insisted of doing main GUI screen and the whole "look and feel" of the software, maximising that consumer would get most modern looking software.

Joshua Bloom

Joshua was assigned the role of programmer. He spent time researching possible implementation techniques. He created a testing platform using google charts. This was to help his team gain an understanding of possible solutions to the problem. This solution was in the form of a HTML google charts visual representation, the program he devolved helped both him and the team gain knowledge of parsing and extracting data. In the implementation phase, Joshua was assigned the task of creating exportation methods. Combined with the aid of ben, Joshua successfully created an exportation system, for PDF, JPEG and HTML exports.

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

Chapter 3 Product Requirements Specification (549 words)

3.I The Team Product SRS update

1.2 Scope [1]

Changed from: "final visual representation can be saved in suitable formats as PDF, DOCX and HTML for EC to read."

Changed to: "final visual representation can be saved in suitable formats as PDF, DOCX, HTML and .txt for EC to read."

Added in .txt format into the program

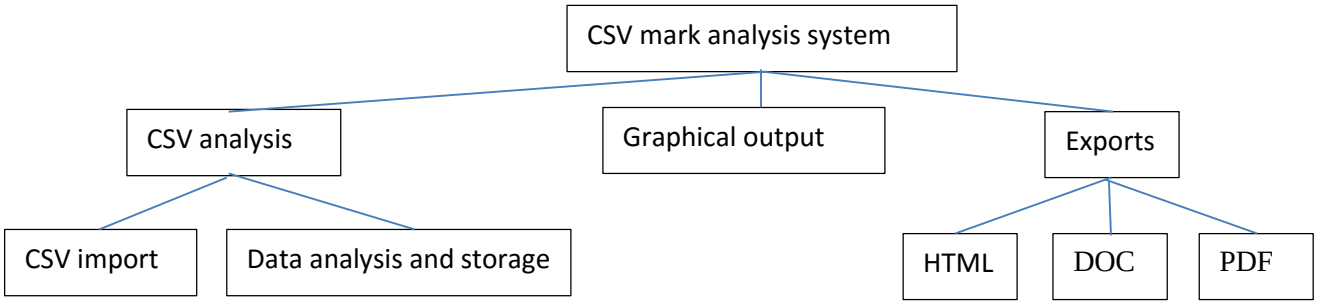
2.1 Product Perspective [1]

Changed from: "and should also provide export functions to .doc, .pdf and also .html formats"

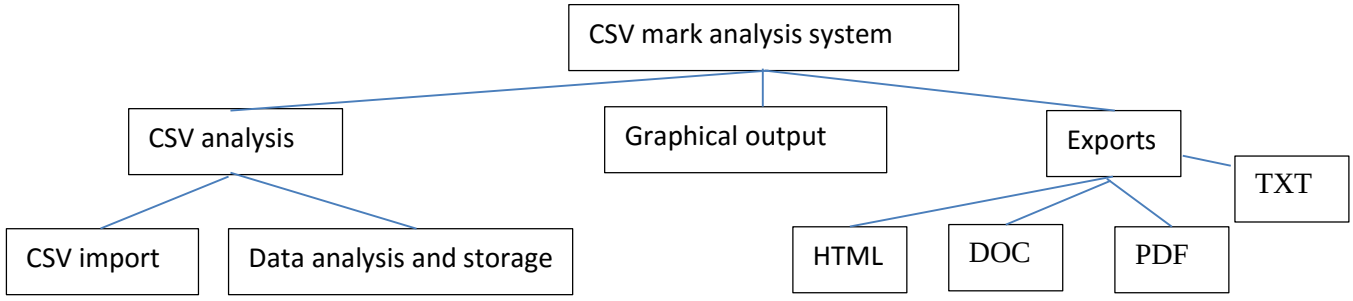
Changed to: "and should also provide export functions to .doc, .pdf, .html and .txt formats"

Added in .txt format into the program

From



To



Added TXT into one of the branches of Exports

System interface

User interfaces

Changed from: "There are no need for other user inputs other than the mouse, and an input to filter the graphs"

Changed to: "The user of the program will need a working mouse and keyboard in order to both select the filter types and .CSV files and to type out what the name of the exported file will be."

Saving file system added to the program requiring the user to type in the file name

Changed from: "They can then also choose an output option, file format and whether it's for students to view or for staff."

Changed to: "The user then has the ability to choose and output option or a file format"

We did not implement a staff or students export function

This paragraph has been removed

Upon execution, a member of staff should have access to the whole program features, as is a student were to get access to the program, they will not have access to the full CSV file, so the security of the system is not of paramount importance.

Due to us not implementing a student and staff security parameter

Software Installation

Added in (Continuing from Adobe PDF viewer): The .txt file can be accessed through any notepad application

We implemented a .txt export function

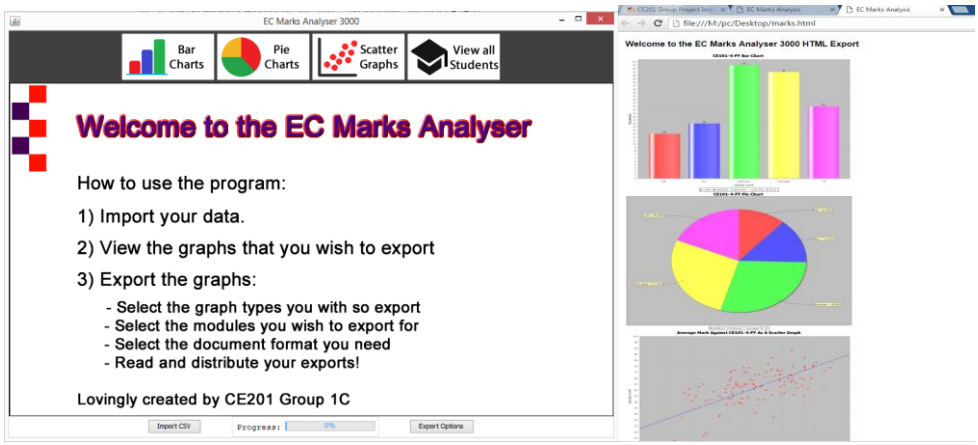
Communication interfaces

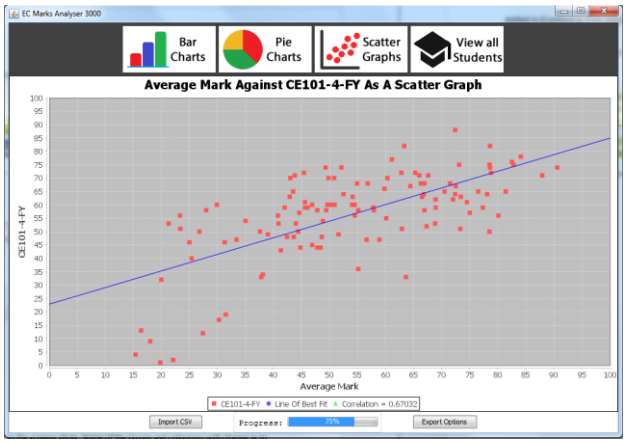
Changed from: "The exports may be shared online via email of FTP, or the HTML file may well be hosted online as a website, seeing that it is of the correct format. This will be catered for when exporting, so no sensitive information will be leaked to the documents. "

Changed to: ""The exports may be shared online via email of FTP, or the HTML file may well be hosted online as a website, seeing that it is of the correct format."

Due us not implementing any sort of encryption or security measure so far

3.1 User interfaces [1]





Images changed due to the program being developed further

3.2 External interfaces [1]

Changed from: "The system outputs will be generated by the system once the user chooses to do so. These will be in the forms of PDF, Doc, or HTML."

Changed to: "The system outputs will be generated by the system once the user chooses to do so. These will be in the forms of PDF, Doc, HTML or .txt.

Due to .txt being implemented in the exportable options

3.II References

[1] Team 1C, "CE201 Team Software Requirements Specification Report", 2016

Chapter 4 Product Development (6333 Words)

4.I Design

Overview

The program specified by the client, the EC committee, is a marks analyser. Its core functionality is to be able to import and handle student's grade data from a CSV file, handle any errors in the data, analyse the data and display a bar chart based on a module, or a scatter graph plotted with the average marks across the X axis, and a chosen module across the Y. The scatter graph has to be plotted with a line of best fit and have a calculated Pearson product-moment correlation coefficient.

The system needs also to export the generated graphs to PDF, DOC and HTML formats. These exports need to be clear and display all the relevant information to enable the users to analyse the graph plots.

We had decided to create our program to use a Java GUI. The GUI should be simple and intuitive for users to pick up and benefit from the full functionality of the system. It should also have a simple guide on how to use the system. There should be a set of buttons that allow the users to import data, export data, and view the graphs. We have decided to include a data table view too, which enables the users to view the data on a larger scale, so they can see specific student's marks.

The entire system should have a clear and formatted styling, with white backgrounds for the data, standout buttons and a progress bar which enables users to view how far they are into the whole export process. We decided to stick to this, with a darker header bar, which makes the image buttons stand out even more. All out exports are formatted to the same style, with a white background and clear titling.

We plan to use the Java JFileChooser to enable file selection and input. This will enable the end user to select the specific files, specify directories and file names, and also enables us to filter files that are viewable, which will enable a higher level of control over the files available for import.

During file exportation, the user should be able to pick any graph, any module and any format to export to. The clearest way to do this is in the form of checkboxes. These enable a clear and constant view of the available options for the user to choose from.

The exports will be formatted in a similar fashion, with a title, and an error message if necessary, with the graphs below in a clear and scrollable format. There should be 2 graphs to a page, and on the webpage should be stored locally in a large resolution. The TXT file output should contain all the student's marks on a single line, with all students accounted for.

The program should be coded in a self-explanatory format. It should contain logically named variables, methods and classes. There should be a main class, in which the GUI is stored and all control of execution flows through, as it is the single interface with the user. Program structure should be modular, so that it is readable and can be expanded in the future. The program should be bug free for release, and have error handling and should notify the end user of any problems that occur, and handle them gracefully without causing difficulty for the user. It should also be secure and not export sensitive student data.

Design constraints

There are some design constraints that have been assessed and been decided upon by the team before development had started. The first is the functionality. The program must not have any bugs on release. Test driven development will be used to allow for a single set of final tests to ensure there are no problems. If there were any bugs, the program, that will process potentially sensitive data, could put that information at risk. The second constraint is operating system based. We plan to have the system functional on Windows, Mac and Linux, provided that the devices Java virtual machine is up to date. Finally, the program must have a professional look and feel. The program must be 'skinned' appropriately to increase the professionality.

Modules

The systems design is to be as modular as possible. This means that each graph should have its own class, exports are combined into a class, the different windows are kept separate from each other and the student format has its own class.

This design allows us to work on separate objects at the same time, and combine them without error. We can also pinpoint errors to separate class files, so that we can debug easily. This modular design is efficient and practical for group work.

There will be seven main classes: GUI, Exports, BarChart, PieChart, ScatterGraph, ExportOptionPane and Student. These will all hold their own methods and variables that are related to the operation of the objects function.

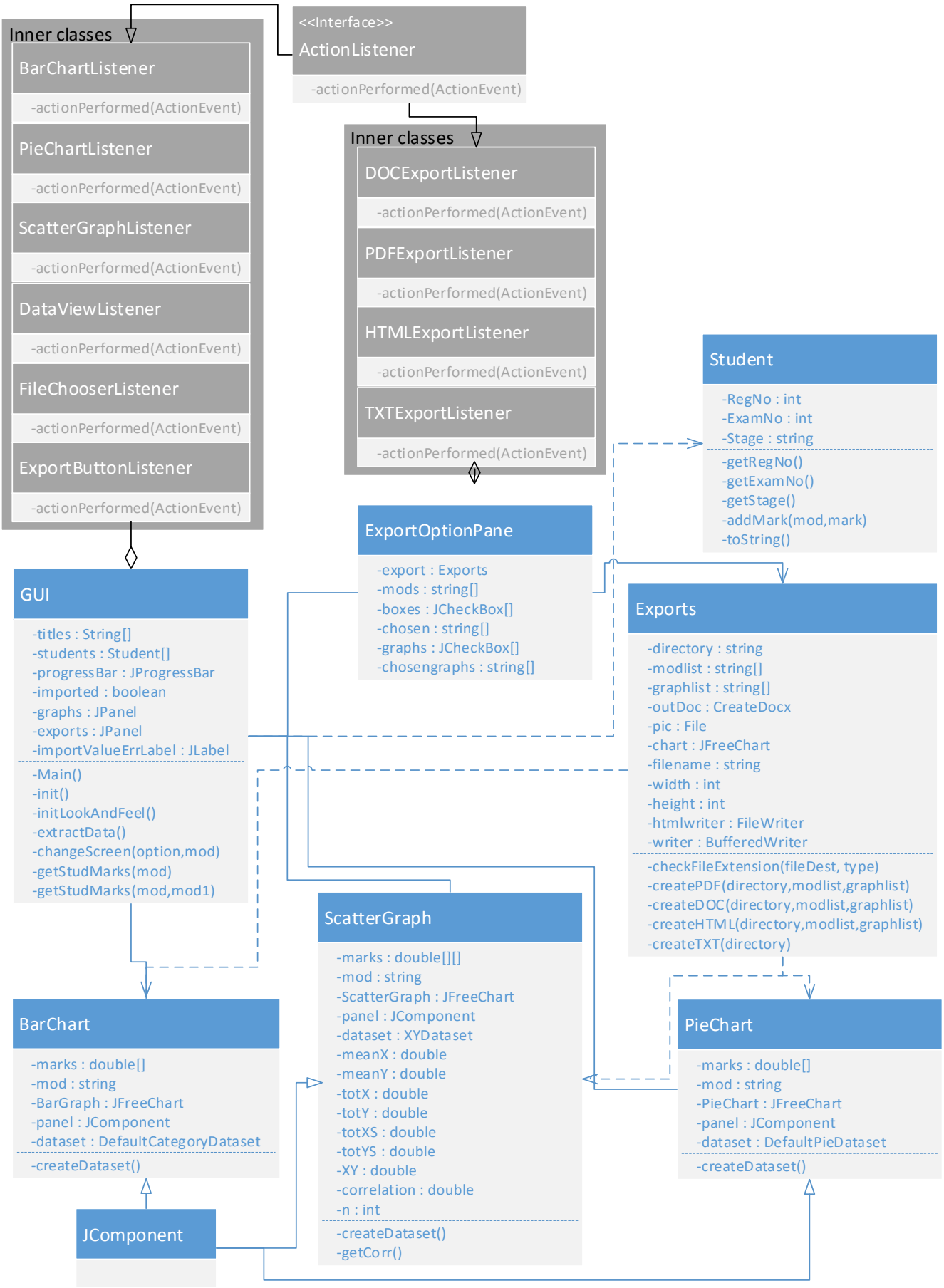
The GUI class will contain all of the main methods for data import, parsing and collecting data from student objects, the main use of the GUI including all buttons and it should cater for and alter the display on the middle of the GUI. This should be the same for the ExportOptionPane. The methods to create exports should be contained within this class.

The BarChart, ScatterGraph and PieChart classes will contain methods and variables relative to the generation of the graphs. This will allow us to generate graphs easily and to refer to graphs as objects. The calculations for the line of best fit and Pearson product-moment correlation coefficient should also be stored here.

The Exports class should contain methods relating to the exportation of the reports. There should be no global variables, but instead, many local method variables, as each export will be different. This may eliminate any errors when producing many exports.

The idea of using many objects is to achieve a low coupling. This enables the classes and methods to function properly when all they know is the data type they will receive. Any methods that are core will be stored within the main GUI class, the rest will be distributed up into their appropriate classes. This design construct will also allow for high cohesion, as the methods will return only a specified data type that is niche and is only called when needed for specific jobs, such as getting student data to create graphs. All that is needed is mark data, so only this will be returned.

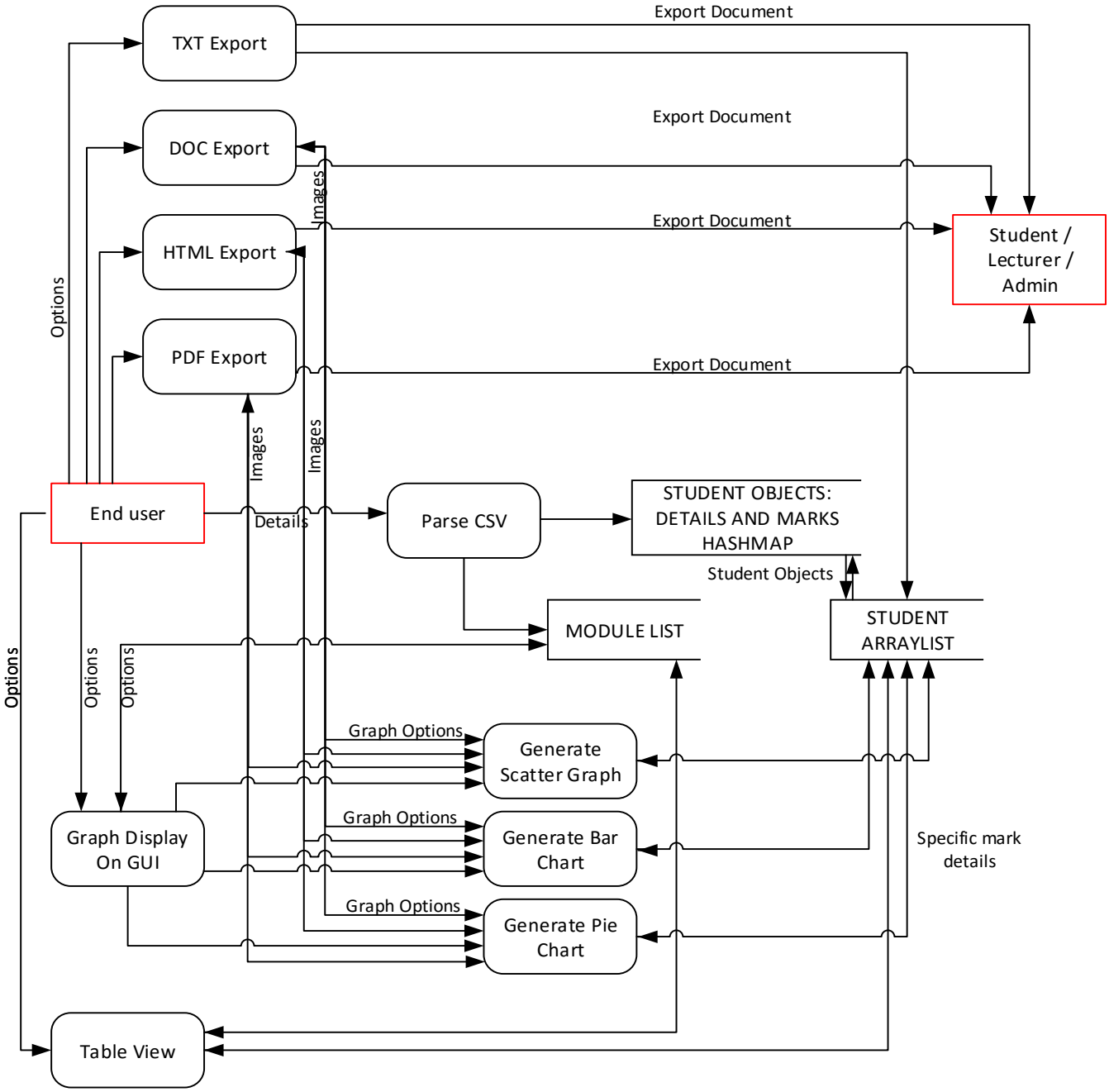
The class structure will allow for good encapsulation and a high level of information hiding. If the graphs are updated, only these modules will change. This separates interface from implementation.



Detailed class Diagram of proposed system.

Data Structure and flow

The aim of the system is to be able to import a CSV file that contains student marks, and for the system to automatically parse this data and store it. There will be many data structures, but the main structure will be an ArrayList containing all the student details. This will be formed when the CSV is parsed. The method will create an array containing all the module codes and store it within the main GUI class, and create a Student object for each line of the CSV that follows. The Student object will contain the Registration Number, Exam Number and stage for each student. It will also contain a hash map that will hold all student marks. This has been chosen because it will provide fast and simple access to the data when we come to collect and search for marks for graph creation



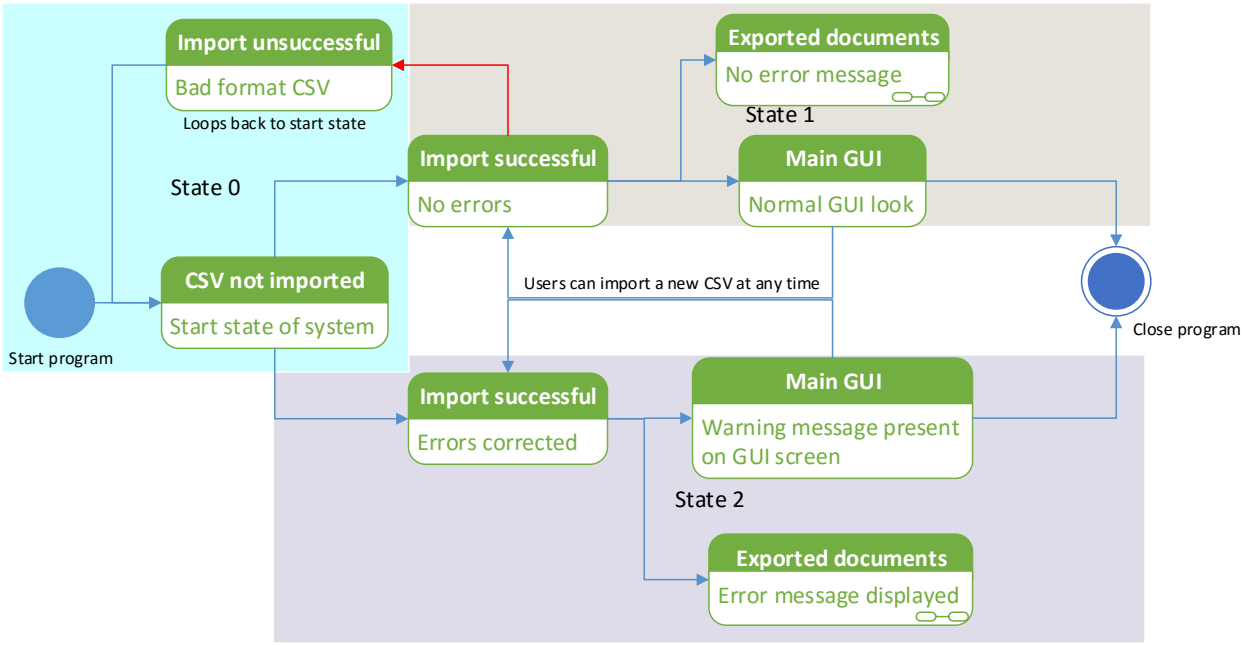
Data Flow Diagram. Shows flow of data between main system functions and data stores.

User error and fault tolerance

The program will have automatic fault detection upon CSV import. This will be in the parsing stage. When the CSV is parsed, the format will first be checked. If there are any errors, a message should be displayed to the user, and they will be able to re try with another CSV. This will be done by checking the first line of the CSV which would be stored in the titles array in the GUI class.

The second form of error checking will be done when creating student objects. When the data for each student is read in from the CSV, the values should be checked. If the mark values do not fall within a 0-100 range, or are not valid numbers, the marks should be disregarded and a message displayed to the end user once the CSV has been imported. This should also mean that a constant error message is displayed on the GUI and in the exports, warning of a missing input.

This will leave the system in one of three states throughout execution.



State diagram. Shows one of three possible program states.

Security

Security is an element of the system that will developed. The data that is imported, will be readable by the user, so that sensitive data is already accessible to anyone. This must mean that the user of the computer is privileged enough to access sensitive data. This enables the table view to be a viable feature of the system. The worry about security is therefore related to the exports.

The exports from our program will contain no sensitive information at all. This is ensured by the storage and retrieval methods used. All student data will be stored as Student objects in a larger ArrayList located in the main GUI class. Storing the mark data in a hash map, allows the safe storage of data for each student, and the variables in this class should be kept private. They should be referred to by using getters, and the key data (RegNo, ExamNo and Stage) should be set on start.

The exports will use the getStudMarks methods. These methods in turn access only getter methods for the data, and will not return any sensitive information other than the marks. The marks will then be entered into the graphs in a random and anonymous format, making the exports secure.

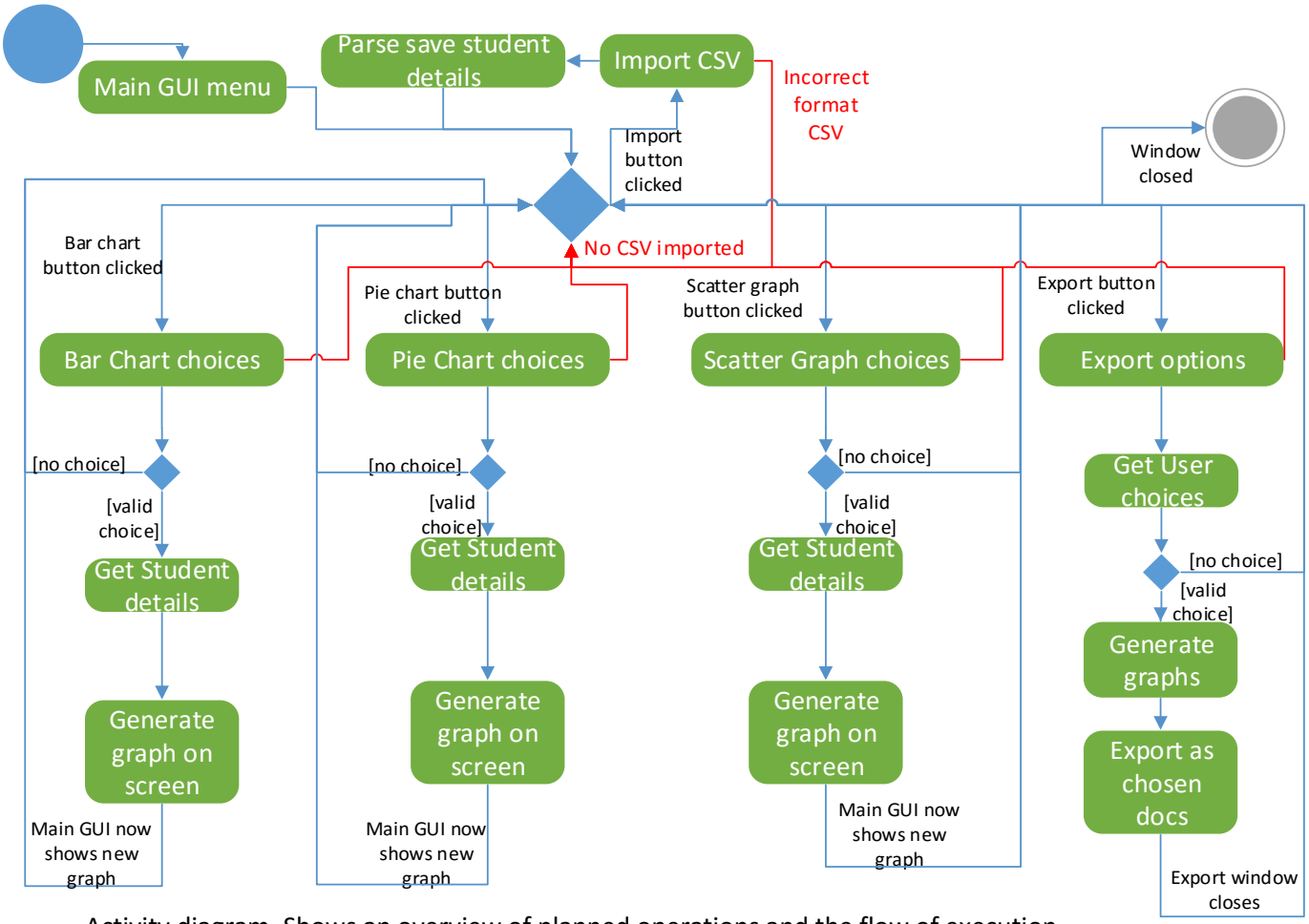
Events, concurrency, and control flow

Control of our system will flow between different methods and classes dependant on the users choices. The user will be presented with multiple options after a successful CSV import. Their choices should then control the processing.

The user will be able to use their mouse to navigate and their keyboard to input file names for the export system.

The facilitator class for all events and diversions of control flow is the main GUI class. This will be constantly running and use listeners to listen for button presses and user selection.

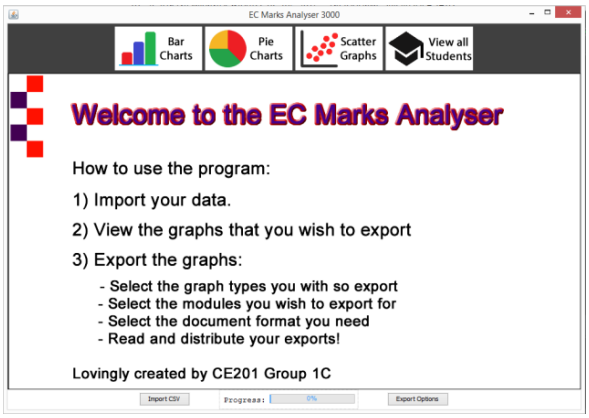
Java alert boxes will be used to give the user any error or success messages, option panes will allow for simple input in the form of drop down menus when the user selects modules to view, and a custom jframe will allow for user input when exporting. These windows will run at the same time as the main GUI, and the ExportOptionPane should not cancel out use of the main GUI. This would be helpful if the user wanted to preview some graphs at the same time as choosing exports.



Activity diagram. Shows an overview of planned operations and the flow of execution.

4.II Implementation

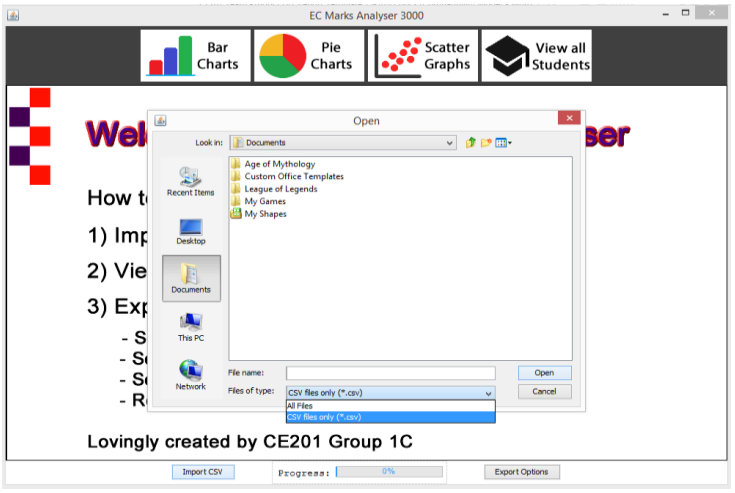
The implementation of the software was written as described in design solution, using OOP structure for fine trouble shooting. We started with a GUI which would hold most of the data rendering. In the first SRS document there are pictures of first prototype version in 3.1 section, which was fine, but we fine-tuned some details like top bar buttons, moved import button to the down bar of the GUI as well as making export button more refined with export pane class.



We only encountered one mayor design change, it is to change middle panel “graphs”. Initially we wanted to make it as a guide for users, but research showed that if we want to stick to JPanel we will need to over write a JPanel class which we didn’t want to do as it is very time consuming and it would increase complexity, so we decided to go with JLabel, because either way we load pictures to a middle part of the screen. With this change product functions stayed the same and nothing more needed to be changed to GUI itself. Next we clearly stated in a design that we want to make GUI more user friendly so we made that software would detect on what kind of OS software is running on and from given system name it would change look and feel of the product, this gives a product familiar feel to whole operating system.

```
private static void initLookAndFeel(){
    String lookAndFeel = null;
    if(LOOKANDFEEL != null){
        if(LOOKANDFEEL.equals("Metal")){
            lookAndFeel = UIManager.getCrossPlatformLookAndFeelClassName();
        }
        else if(LOOKANDFEEL.equals("System")){
            lookAndFeel = UIManager.getSystemLookAndFeelClassName();
        }
        try{
            UIManager.setLookAndFeel(lookAndFeel);
        }
        catch (Exception e){
            System.err.println("");
        }
    }
}
```

The first thing that GUI class asks is to import data which must be in CSV format, we implemented JFileChooser for user to pick up a raw data file and statically made that only .csv format can be taken, this prevents users to load any other files to the software.



After a file is successfully chosen CSV Reader read the file and checks if a file in correct template format. It checks first three cells in csv if they are correct, student object is created, and more is covered in error handling section. Student object is an array list which holds all the students, in format that first three csv cells are student registration number, exam number and stage followed by modules student has marks in, marks are stored in hash map. During data extraction we parse all marks to double format as we assume students get very detailed marks. Additionally we recalculate average student marks just for proof checking.

Secondly, the GUI mainly operates on changescreen() method which changes middle section of the software. Below you can see a part of the code, the only difference is that each case calls out different button.

```
public static void changeScreen(String option, String mod) {
    graphs.removeAll();
    switch(option){

        //if chosen graph is bar
        case "bar":
            BarChart barchart = null;
            //plot average marks
            if(mod=="Average Marks"){
                barchart = new BarChart(getStudMarks("TC_MK"),mod);
            }
            //plot another module
            else {
                mod = mod.substring(0,10);
                barchart = new BarChart(getStudMarks(mod),mod);
            }
            graphs.add(barchart.panel);
            progressBar.setValue(75);
            break;
```

One thing that was not mentioned in our SRS is implementation is the “View all Students” feature, which allows users to view parsed information of CSV file. It uses JTable component with JScrollPane, this lets users to have full view of the document and it is not limited just by GUI screen size. Implementing this feature was part of error handling, because it allows users to view full data that software is using to make visual representation. Bellow you can see the code for “View all Students” feature. At the bottom of the code we added graphs.repaint() method that each time middle part of the GUI is repainted and neat and interactive way is remained consistent.

```
    case "dataview":
        Object[][]studs = new Object[students.size()][titles.length];
        for(int l=0;l<students.size();l++) {
            studs[l][0] = students.get(l).getRegNo();
            studs[l][1] = students.get(l).getExamNo();
            studs[l][2] = students.get(l).getStage();
            for (int m=3;m<titles.length;m++){
                studs[l][m] = students.get(l).marks.get(titles[m]);
            }
        }
        JTable table = new JTable(studs,titles);
        JScrollPane tableScreen = new JScrollPane(table);
        tableScreen.setBounds(1,1,graphs.getWidth(),graphs.getHeight());
        table.setBounds(1, 1, graphs.getWidth(), graphs.getHeight());
        table.setVisible(true);
        tableScreen.setVisible(true);
        graphs.add(tableScreen, BorderLayout.CENTER);
        break;
}
```

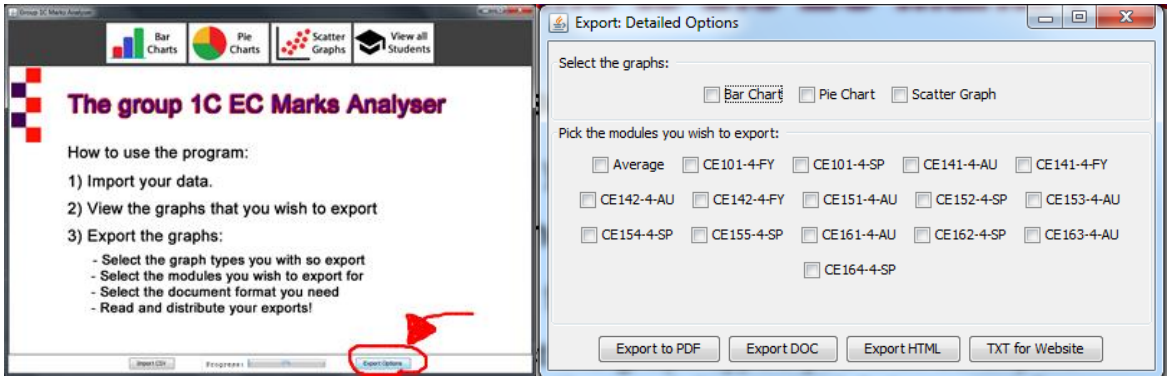
Another feature that we implemented to GUI is “Progress bar”, it is made just as design states, which is to show how much progress user is into our software. Progress bar uses static values at each step of the user. Ex: Successful import is 50 % viewing visual representation 75 % and finally exporting sets you to 100 % showing that user used core functionality of the software.
Of course GUI class has many more methods like button listeners, they all do the same and the only difference is that they use selected switch case which triggers changescreen() method.

Exporting:

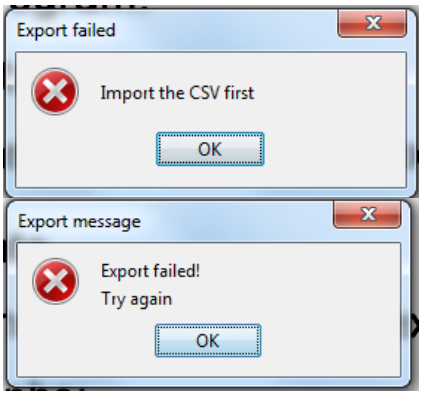
The exportation of the visualized data was one of the fundamental features that the software had to incorporate. The exportation mediums we used were: (doc,pdf,html,txt).

The SRS stated we only required one exportation method, however we knew that the additional exports would greatly increase functionality and portability of our system and this is why we spent the time in developing the additional exportations of the visualized data.

The exportation system is easily usable. It is first accessed by pressing the ‘export’ button. This prompts the user with a choice of 4 options (doc,pdf,html,txt) the user simply selects an option and choses a file location. Once the location has been chosen the exportation function will begin.



Errors:



As a precursor, if no CSV file has been given for the parser to read, another prompt will also be thrown to the user. Stating to give a csv file first.

Please note that if any errors occur with the export, an exception is thrown and the user is issued with a message. The export selection will not have changed though.

Once the exportation function has been completed successfully. A new file will appear of the designated export type and in the chosen file location with the correct filename.

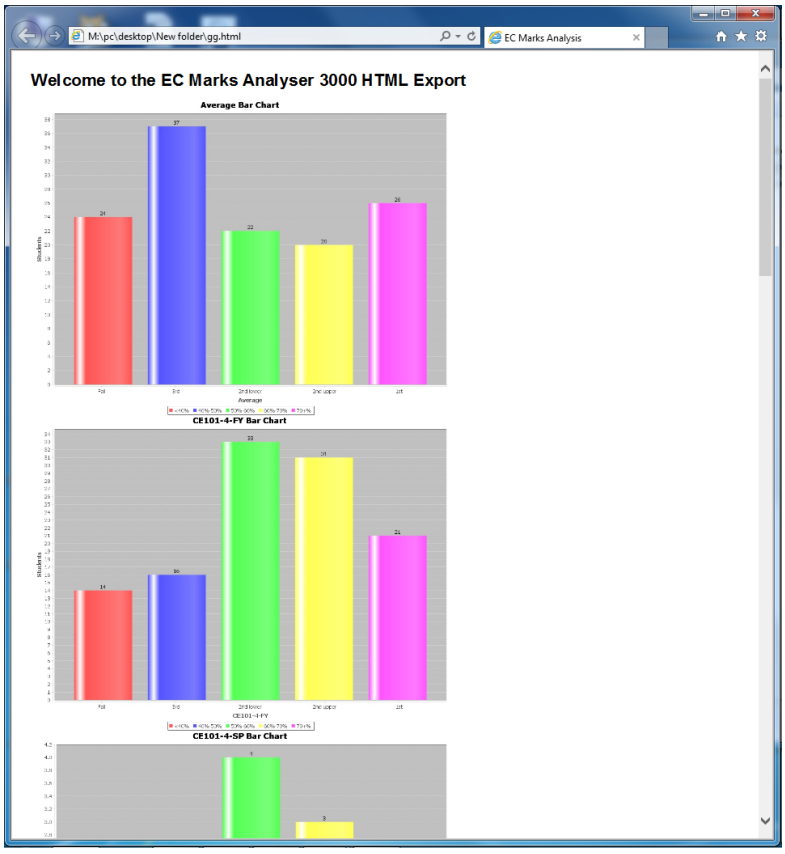
The html export works a little differently however, the user can export multiple visulization to one html file. The images are placed within a simple HTML table with a set width.

When the images are exported, a simple piece of HTML code is written to the file.

The images are presented in a neat, consise and formated manner.

Simple code written to the file:

```
<html>
<head>
</head>
<body leftmargin=36.0 rightmargin=36.0 topmargin=36.0 bottommargin=36.0>
<img src='M:\WORK\groupResource\svn\ce201-03\22-02-16\images\CE162-4-SPBar
Chart.jpeg' width='800.0' height='600.0' /></img></br>
</body></html>
```



The exported image, are of jpeg format. This is the best format for websites. And technically its another exportation option because the user can save the jpeg images from the website to their system.

The final export options ‘textHTML’ this is a txt file that is created for use with our auxilary online system. This system will be explained later in the chapter.

PDF export implementation:

PDF is a very widely adpoted file format, that is used by many people and compatiable with almost every device. Therefore it was very important for us to make sure that our system can export to this file type.

We conducted some research into the best way of exporting our data to pdf. We found a very useful Java libary called 'iText' Itext made the exportation of the PDF relatively simple.

Firstly we create a new outputstream in the form of a file, this file will become the basis to which all the data is streamed.

With several if statements, we can find out what type of chart has been given for exportation.

(bar,pie,scatter). We then 'create' a new chart of this type.

Once the chart has been created it is streamed into an image form, this image is then given a hieght and width.

The image is also scaled, so that the quaility will be high making the image easy to read once exported to PDF.

Once all this has been complete, the file will be added to the file.

Then the file writer is closed and the exportation is complete.

Here is an example code of a pie chart export

```
else if (graph == "Pie Chart") {
    JFreeChart chart;
    if (module == "Average") {
        chart = new PieChart(GUI.getStudMarks("TC_MK"), module).PieChart;
    } else {
        chart = new PieChart(GUI.getStudMarks(module), module).PieChart;
    }
    BufferedImage bufferedImage = chart.createBufferedImage(width, height);
    Image image = Image.getInstance(writer, bufferedImage, 1.0f);
    image.scaleAbsolute(swidth,sheight);
    document.add(image);
```

reference to iText(http://itextpdf.com/)

DOC Implementation:

The DOC export works in a very similar way to the PDF export.

Firstly we create a new Doc file, like the pdf export.

```
CreateDocx outDoc = new CreateDocx("docx");
```

Also similar to the PDF export, we create a new chart with the given parameters of the chosen chart type.

```
else if (graph == "Pie Chart") {
    JFreeChart chart;
    if (module == "Average") {
        chart = new PieChart(GUI.getStudMarks("TC_MK"), module).PieChart;
    } else {
        chart = new PieChart(GUI.getStudMarks(module), module).PieChart;
    }
    File pic = new File(directory+"\\"+module+graph+".jpeg");
    ChartUtilities.saveChartAsJPEG(pic, chart,width,height);
    HashMap paramsImage = new HashMap();
    paramsImage.put("name",pic.getAbsolutePath());
    paramsImage.put("scaling", "1");
    outDoc.addImage(paramsImage);
```

We create jpeg image and put them in a file, this file is a temporary file that will be deleted once the export has been completed.

Once the chart has been created a hashmap is made, this hashmap is used to define the size and location of the image.

```
HashMap pageSettings = new HashMap();
pageSettings.put("orient", "normal");
pageSettings.put("top", "1000");
pageSettings.put("bottom", "1000");
pageSettings.put("right", "1000");
pageSettings.put("left", "1000");
outDoc.createDocx("Mark Analyser Export",pageSettings);
org.apache.commons.io.FileUtils.deleteDirectory(new File (directory));
```

Finally the images stored in the file are extracted, and place within the newly created doc file, and hey presto the exportation is completed.

HTML Exportation:

Firstly we set a size and a width, these variables correlate with the defined size of the images that will be exported to the html.

```
int width = 1000;
int height = 800;
```

Just like the .DOC exportation, a new folder is created for images. This time however, it stays, as it holds all image folders that are linked in the page.

We then write some very basic HTML code to the html file.

```
FileWriter htmlwriter = new FileWriter(directory.substring(0,directory.length()-
6)+"export.html");
BufferedWriter writer = new BufferedWriter(htmlwriter);
writer.write("<html>\n<head>\n</head>\n");
writer.write("<body leftmargin=36.0 rightmargin=36.0 topmargin=36.0
bottommargin=36.0>\n");
```

This code simply defines the margins of the page, so that when the images are placed within. They look neatly aligned and professionally presented. With these margins each images sits row by row beneath one another. Allowing the user to simply scroll down and see all the presented images.

Like the other 2 exportation methods, will apply several if statements, to gather what type of chart has been submitted for export.

Once we have figured this out, the chart is created and exported as a .jpeg image (similar to DOC) to the temporary file. We then write to the html file again, this time referencing the image located in the temporary file and thus placing the image on the html page using a src tag, like so:

```
File pic = new File(directory+"\\"+module+graph+".jpeg");
ChartUtilities.saveChartAsJPEG(pic, chart,width,height);
writer.write("<img src='"+(directory+"\\"+module+graph+".jpeg")+"' width='800.0'
height='600.0' /></img></br>\n");
```

Notice the 'newline' parameter, this places each image on its own line in the html.

Output formatting:

All outputs are formatted to a clear and consistent manner. They all have headers that reference the output system. They are fitted two graphs to a page on the document exports, and also inform the end user of any errors that occurred during data import.

These features help make out system professional and easy to read.

All graphs are also formatted with headers, being a brief description of what the graph shows, and the axes are labelled and all graphs contain a key.

```
if(GUI.importValueErr){
    errstr = "There was one or more erroneous data items found during import. These
were corrected, and averages recalculated. The resulting data set may be
incomplete.";
}
document.add(new Paragraph("EC Marks Analyser 3000 PDF Export\n"+errstr));
```

This is an example of formatting on the PDF export.

The final exportation option, 'exportTXT'.

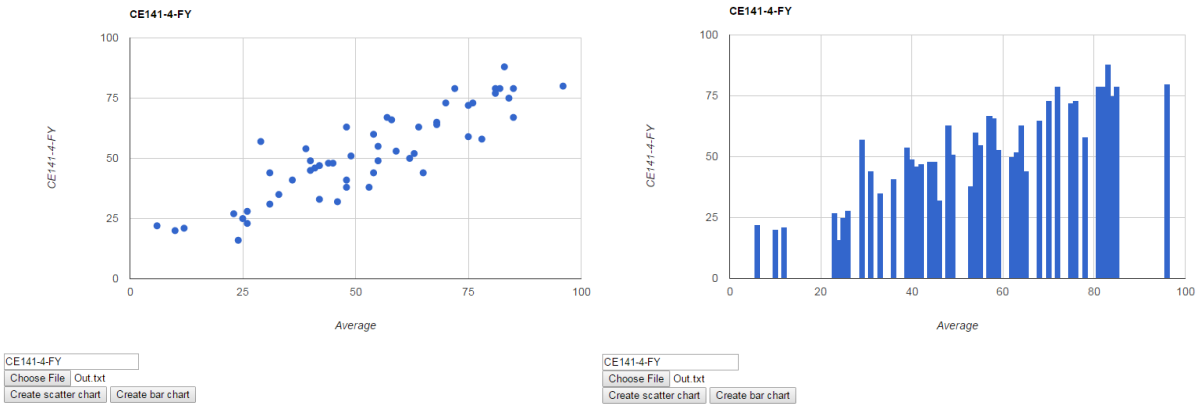
This is very much in testing phase, however if we were to develop the system further. The next stage would be fully implementing this exportation medium.

So what does it do?

Well we have created an auxiliary online system, the online system uses a very neat data visualization library called 'google charts'. Google charts allows for very professional looking modern charts that compute very quickly indeed.

The main logic of the system takes place in JavaScript code, we parse data from a txt file that has been formatted in a way that caters to parsing. We use this data to create the google chart visualization. The ArrayList of students in the main GUI class it accessed, and the toString() methods of the students is used to output all of their mark data line by line, into a txt file. The method uses a simple FileWriter to output.

This txt file is what the 'exportTXT' option creates, the user simply inputs their csv file. And this function will create a txt representation of this csv file, in a format that allows for easy parsing within the JS code of our auxiliary system.



This is an example the graphs generated by the auxiliary system.

Security:

Security was not a major issue in our project, this is because the system is only designed to create visualization of input data. Meaning that the data will only be accessed by the person that inputs it. And this data is obviously already visible to the user. The system is client-sided (offline) so there is no possibility of SQL injection or something like that.

With this noted there is little to mention in terms of security however we have implemented some security features.

Within the java code we have made use of private variables and protected statements, this means that a potential hacker cannot gain access to these variables easily. There are Getter methods, however not setter methods within the student class, so the variables can only be read and not edited.

```
private int RegNo;
private int ExamNo;
private String Stage;
public int getRegNo(){
    return RegNo;
}

public int getExamNo(){
    return ExamNo;
}

public String getStage(){
    return Stage;
}
```

Future Features:

As mentioned before, we have created an auxiliary system. In the future we would develop the main system in this way.

The current system is a Java client-sided software, meaning that it must first be downloaded by the user and initialized correctly. This will also require the user to have java downloaded on their system and an understanding of how to install the system.

If we mounted the system on a server, using similar features already apparent on our auxiliary system. We could eradicate all of the down falls of client-sided software and not lose any functionality in fact it would greatly increase functionality.

If the system is mounted online, no initialization will be necessary. The user would simply enter the URL into their desired web browser and this will instantly redirect the user to our system. This means that the system can be easily accessed/shared between users/clients.

Another great aspect of server sided software, is that maintenance and updates are so much easier.

The user will not need to go and download a new version which could lead to complications. If the system is mounted on a server developers can simply roll out updates, and replace the current version on the server. This will automatically update the system for users and will require no effort on their behalf.

To improve security, some kind of user system would be very beneficial.

Currently we have no system to secure the system form unwanted users, there is also no way to define users. This would enabled us to sensor certain features to certain users. For example, students could use the system to look at their current marks, to assess how they're doing on the course. But we do not want students to be able to view all the marks every student has on the module, and also not allow them to view marks before they have been published.

Using a user system, we can differentiate by some kind of user ID variable and sensor certain features to certain users. A password system, is very essential in protecting data, however returning to my previous point. If this system was mounted online it would be much easier, because we could deploy a database and place every password and user on a secure and encrypted data base that would maintain upmost security.

Error handling:

Throughout our code we have used try and catch statements, we spent a long time contemplating all the possible errors that could occur within runtime.

Users are always notified of any error with an appropriate pop up message. This is every success of export, import of data failing and many more. This means that the end user can get an idea of what the system is doing and why errors are occurring, which may enable them to fix any broken input.

The system also features error detection and limited correction upon input. This happens when the CSV file is parsed by the OpenCSV library.

```
if (titles[0].compareTo("Student RegNo")==1 || titles[1].compareTo("ExNo")==1 ||
titles[2].compareTo("Stage")==1 ){
    JOptionPane.showMessageDialog(window, "Error with CSV structure.\nTry again
with the correct CSV.", "CSV Error Notification", JOptionPane.ERROR_MESSAGE);
    titles=null;
    return false;
}
```

This shows the error detection in file format, and hoe the user it notified.

Also, in the main student object creation loop, where data is parsed into the system, any errorneous data is ignored and the averages recalculated.

```
if(Double.parseDouble(nextLine[x])>100.0 || Double.parseDouble(nextLine[x])<0.0){
    importValueErr = true;
}
```

```
if(importValueErr){
    JOptionPane.showMessageDialog(window, "Error with one or more student
marks.\nError was rectified during import\nand mark was not imported.", "Import
Error Notification", JOptionPane.WARNING_MESSAGE);
}
```

The system still imports the remaining data, however a warning message is shown on the GUI and in reports.

4.III Testing

Testing

There were many tests planned by the programmers, which were designed to test every element of the system. The tests were aimed at both normal everyday use, and also at the possible weak points with CSV formatting and error detection.

The tests were laid out into 4 series. These included detailed testing into: the loading and GUI screens; the Import and parsing of CSV files; the Graph and table displays; and the exports.

Provided within the testing documentation is a detailed list of tests, with the practice for the tests, expected result, the received result and screenshot evidence of the result. This enables the supervisors or other people in the development team, if necessary, to replicate tests. All testing was done with the dummy CSV file available on the CE201 Moodle pages, (it is also linked in the documentation).

Series ID	Area of system to be tested
1	Loading and GUI screens
2	Import and parsing of CSV
3	Graph and table display
4	Exports

CSV file that was used for testing:
https://moodle.essex.ac.uk/mod/book/view.php?id=279190&chapterid=5179

Test ID	Test practice	Expected result	Received result	Evidence ID
1.1	Start program	Main GUI screen loads	When you start the program the GUI screen loads up	1
1.2	Click on every button apart from import	Error message should display	If you select any button on the GUI there will be no response due to no csv file being inputted	2
1.3	Select import	import file chooser displays	When you select import the file chooser is displayed	3

1.4	Click on export button	Export window displays	When you click on the export another window is displayed on the screen requiring what to save the file as and what location	4
1.5	Click on Bar chart button and make choice CE101-4-FY	Bar chart popup displays and then displays bar chart	When you click on the bar chart button and make the choice for CE101-4-FY, the bar chart is displayed on the GUI	5
1.6	Click on Pie chart button and make choice CE101-4-FY	Pie chart pop up displays and then displays pie chart	When you click on the pie chart button and make the choice for CE101-4-FY, the pie chart is displayed on the GUI	6
1.7	Click on Scatter graph button and make choice CE101-4-FY	Scatter graph pop up displays and then displays Scatter graph with Correlation = 0.7851	When you click on the scatter graph option it produces a drop down list and once you click on CE101-4-FY it produces a scatter graph of that particular module, however the correlation is equal to 0.67032 rather than 0.7851	7
1.8	Click on data view button	Data table displays all results and user can scroll	When you click on the data view button it shows all the students, modules and grades	8
1.9	Close window	Window closes	When you click on close it closes the GUI	9
2.1	Import dummy csv from	CSV imports and success pop-up displays	When you import the dummy CSV file it imports it and shows a pop up box showing that the importation was successful	10
2.2	Import incorrect format CSV – change Student reg no field on dummy CSV	CSV error pop up displays and user sent back to home screen, but cannot click any buttons still	CSV error doesn't pop up however, GUI shows no response and requires a valid csv with the student reg no filled in	11
2.3	Import CSV with an incorrect data item – change first CE101-4-FY mark to 999	CSV error pop up displays regarding the import error, been corrected. Success window also pops up and a error message at the bottom of the GUI shows throughout use. Users mark will be removed. Data view will not show the mark.	When you input an invalid number such as '999' the GUI will still import this csv file, however, it will remove this mark from the csv file and display an error message at the bottom of the GUI	12
3.1	Select to view bar chart from CE101-4-FY	Bar chart should display with a key and numbers above the bars. Bars are	When you select to view the bar chart from CE101-4-FY it will	13

		for each grade but labelled with marks not degree class	display this on the GUI including the key, numbers and grades	
3.2	Select to view pie chart for CE101-4-FY	Pie chart should display with percentages for each class of degree	When you select to view the pie chart from CE101-4-FY it will display this on the GUI including the percentages for each class of degree	14
3.3	Select to view scatter graph for CE101-4-FY	Scatter graph shows with all points plotted. Average as X and CE101 as Y. Key displays the lines and the correlation which should be equal to	When you select to view the scatter graph from CE101-4-FY this will be shown on the GUI, including the average, key and correlation	15
3.4	Select to view the table view for all data	Table will be shown full screen. With working scroll bar (if enough data)	When you select the table view this will output the entire csv file on the GUI	16
4.1.1	Choose to export PDF of bar, scatter and pie charts containing module CE101-4-FY and Average. Select name PDFTEST	User presented with pop up file explorer. Enters name and clicks save. Success pop up shown. Export will be in location, formatted with a header and showing the graphs two to a page.	When you choose to export the csv file to contain all the type of graphs and the CE101-4-FY and average scores, this will export a file named 'PDFTEST' onto the specific location you want	17
4.1.2	Choose to export PDF of bar, scatter and pie charts containing module CE101-4-FY and Average. Select name PDFTESTFAULTY (Must use data that had a bad value that was ignored)	User presented with pop up file explorer. Enters name and clicks save. Success pop up shown. Export will be in location, formatted with a header and an error message and showing the graphs two to a page.	If you were to import the csv file which contained the error within, the GUI will still successfully import this file, then export this csv to a pdf file. However, when its exported it will show an error message in the header of the file	18
4.2.1	Choose to export DOC of bar, scatter and pie charts containing module CE101-4-FY and Average.	User presented with pop up file explorer. Enters name and clicks save. Success pop up shown. Export will be in location, formatted with a header and showing the graphs two to a page.	When you choose to export the csv file to contain all the type of graphs and the CE101-4-FY and average scores, this will export a file named 'DOCTEST' onto the specific location you want	19

	Select name DOCTEST			
4.2.2	Choose to export DOC of bar, scatter and pie charts containing module CE101-4-FY and Average. Select name DOCTESTFAULTY (Must use data that had a bad value that was ignored)	User presented with pop up file explorer. Enters name and clicks save. Success pop up shown. Export will be in location, formatted with a header and an error message and showing the graphs two to a page.	If you were to import the csv file which contained the error within, the GUI will still successfully import this file, then export this csv to a doc file. However, when its exported it will show an error message in the footer of the file	20
4.3.1	Choose to export HTML of bar, scatter and pie charts containing module CE101-4-FY and Average. Select name HTMLTEST	User presented with pop up file explorer. Enters name and clicks save. Success pop up shown. Export will be in location with local images folder containing all graphs, HTML is formatted with a header and the relative graphs below.	When you choose to export the csv file to contain all the type of graphs and the CE101-4-FY and average scores, this will export a file named 'HTMLTEST' onto the specific location you want	21
4.3.2	Choose to export HTML of bar, scatter and pie charts containing module CE101-4-FY and Average. Select name HTMLTESTFAULTY (Must use data that had a bad value that was ignored)	User presented with pop up file explorer. Enters name and clicks save. Success pop up shown. Export will be in location with local images folder containing all graphs, HTML is formatted with a header and error message displayed and the relative graphs below.	If you were to import the csv file which contained the error within, the GUI will still successfully import this file, then export this csv to a html file. However, when its exported it will show an error message in the header of the html	22
4.4.1	Choose to export TXT file,(does not matter about selected graphs or modules) name it TXTTEST	User is presented with a TXT file that contains all student data as plaintext. Student regno, then each module and mark as (Module|Mark) format. Each student on own line	When you choose to export the csv file to contain all the type of graphs and the CE101-4-FY and average scores, this will export a file named 'TXTTEST' onto the specific location you want	23

Chapter 5: Project Management (555 words)

5.I Project Management Report

Methodology and approach

The product was developed within an agile-like environment, with a heavy focus on test-driven development. This allowed us to output a new iteration of the product, test it, and move on. This has been a great aid in development as there were a few challenges regarding the program that needed to be met before we proceeded.

All tasks assigned were met swiftly and well within the given timeframe. This was an early concern regarding the final product, as we seemed to meet a few challenges and the development plateaued for a brief time, this was soon moved past, however, and has not impacted the final deadline.

Final Product

The product was finally finished and fully tested on Friday the 11th of March, in time to be presented to the customer on Monday the 14th.The presentation was created on Monday the 7th, revised on Friday the 11th, and a final copy uploaded Monday the 14th. The product and presentation were delivered on time and according to schedule. This is in no doubt aided by the test-driven development environment, allowing us to cut back time on final testing and debugging by increasing the time taken for the implementation.

5.I.a The team Gantt charts (See Appendix 2)

5.I.b A discussion of the Gantt charts

For the most part, the Gantt charts do appear fairly similar overall, however, it is evident that there has been some restructuring with the assigned tasks. The first chart only included the tasks associated with the team report and not the final product, this was a mistake that was rectified in the second chart. It is also evident that with the second chart, much more detail has been included. Having only the bare essentials in the first chart did prove troublesome as they were much too large and far too vague to follow directly. The overall structure of the first chart, however, proved to be a good representation for the overall timescale that we were facing (even if the task lengths were perhaps incorrectly estimated). Note in the second Gantt chart, many of the tasks were placed in sequence, due to various dependencies, this includes the final testing, which could not be completed until the product was fully implemented. With regards to testing however, we built the product within a test-driven development environment. This allowed us to test the product “on the fly”, giving us the opportunity to scale back the final testing period, as it could be ensured that the code was functional before a large final test cycle. This helped greatly, as it can be seen that our implementation section in the second chart overran that which had been estimated in the first chart.

Had the first chart been more in depth, it is likely it would have more closely resembled the second in terms of general structure. The iterative development cycle we implemented was definitely a wise move in regard to testing and debugging.

5.II.c An evaluation of the project management

Overall, team progress was very good. The team worked as a cohesive unit, with everyone being kept up to date with product progress and pitching in where they could. It was evident early on that some had much more experience with coding, and they were put to the front in regards of developing the product. All team members, however, did have input in regards to the design as well as other tasks assigned which were relevant to the deliverable.

Chapter 6: Conclusions (621 words)

On the whole, we feel that the project went well. Everyone is happy with the final outcome of the system, and people feel that it exceeded some expectations. We worked as a team, and all contributed within the lab sessions to the program, whether it be in project organisation, ideas, code, or testing.

The system was designed for the education committee of the CSEE department, and is designed as a tool for the analysis or a large collection of student marks. Our system does its job successfully. It imports and stores a potentially large collection of student details, and is able to process them quickly, using hashmaps to store the mark data, and expandable arrays of students. The reports that are generated are completely customisable and are output in a professional and clear format.

We included error detection that relates to general use, informing the users of errors as they occur. The system also informs and corrects errors regarding the input of the CSV file. We made sure that the system would be accurate, so recalculating the average marks was the best option here.

The system was also programmed with an interactive GUI, allowing users to view the visualisations of the data before export, and we created an intuitive and modern looking GUI to match this.

We produce more than the required amount of graphs, as we also implemented a pie chart to show percentages, and also a table view, so data can be viewed on the go. The graphs that we produce are up to standard. They show whatever module that the user requests, and the scatter graph plots the module against the average grade of the student, the line of best fit, and calculates and shows the Pearson's product-moment correlation coefficient.

The system meets and exceeds the coding requirements of the project. It incorporates good coding practices such as modularity, inheritance and interfacing. The design of the system is that of a high cohesion and low coupling, enabling errors to be contained within subsystems, easy debugging and easy extension for future. There is a high level of information hiding, so the end user is presented with a clear and concise interface. The class structure also allows for good encapsulation of data and methods.

The project management was good, and we time planned effectively. The jobs may have not been distributed as effectively as possible, however, group members always had something to do, whether it related to the actual program, or the documentation that runs alongside it. The project manager did a good job at keeping the Gantt chart up to date, and informed us of deadlines in meetings. The minute's keeper also did a good job at tracking what we spoke about and planned in each meeting. These elements allowed for a streamlined development process.

We planned some additions to the program for the future, and even began developing them alongside the normal system. We feel that, had we been given the chance to implement them, the system would have another level of usability, which would benefit the users largely.

If we were to undertake this project again, there are some things that we would plan differently. The distribution of jobs would change a lot, and the whole programming team would become more involved, developing classes separately, then combining them. This would have aided development load and decreased the time taken. We would also choose to implement more features, such as a user verification mode, student filtering and a possible online mode. Our error checking would also be more robust and allow users to fix errors. We would have also dedicated more time to the original SRS document, to improve our plans which would have aided development a lot.

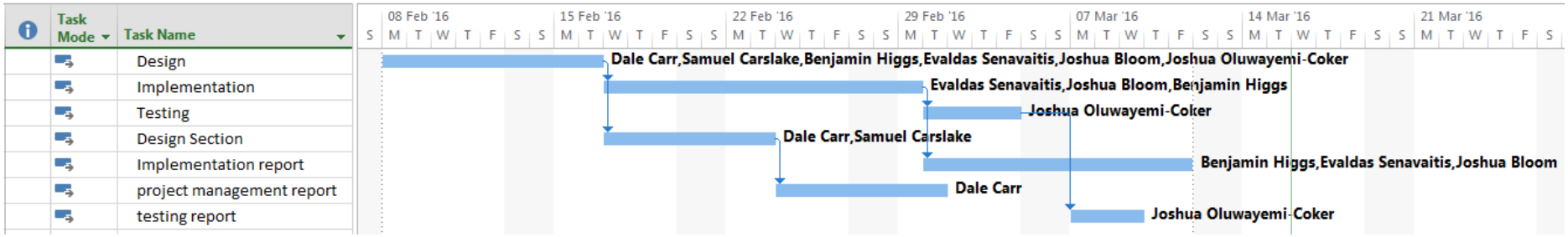
Appendix 1: The Team Effort Summary Table

Member		Joshua bloom	Evaldas Senavaitis	Samuel Carslake	Benjamin Higgs	Dale Carr	Joshua Oluyawemi-Coker
Role			Technical lead			Project Manager	
Chapter	**Sub-section**						
Executive summary				I	I	I	
Team Working	Team activity Report	I	I	I	I	I	I
SRS				R			
				R			
Product Development	Design		C		R		
	Implementation	I	I		C		
	Testing				I		I
Project Management	Report					R	
	Evaluation					R	
	Gantt Charts					R	
Conclusions		C	C	C	I	I	C

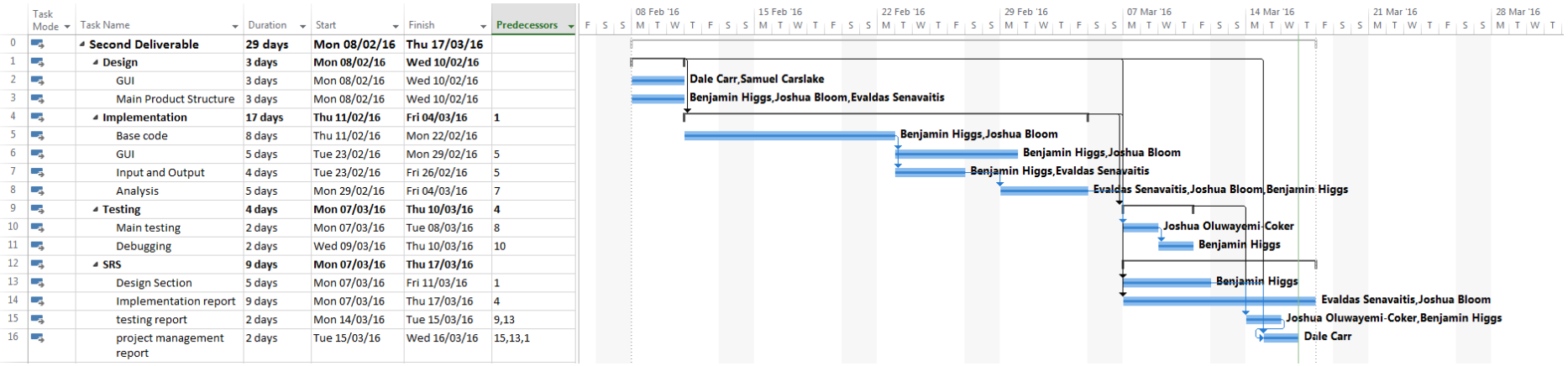
Complete the set of tasks performed for this report and indicate R=Responsible, I=Involved, C=Consulted

Appendix 2: The Team Gantt Charts

First Gantt chart

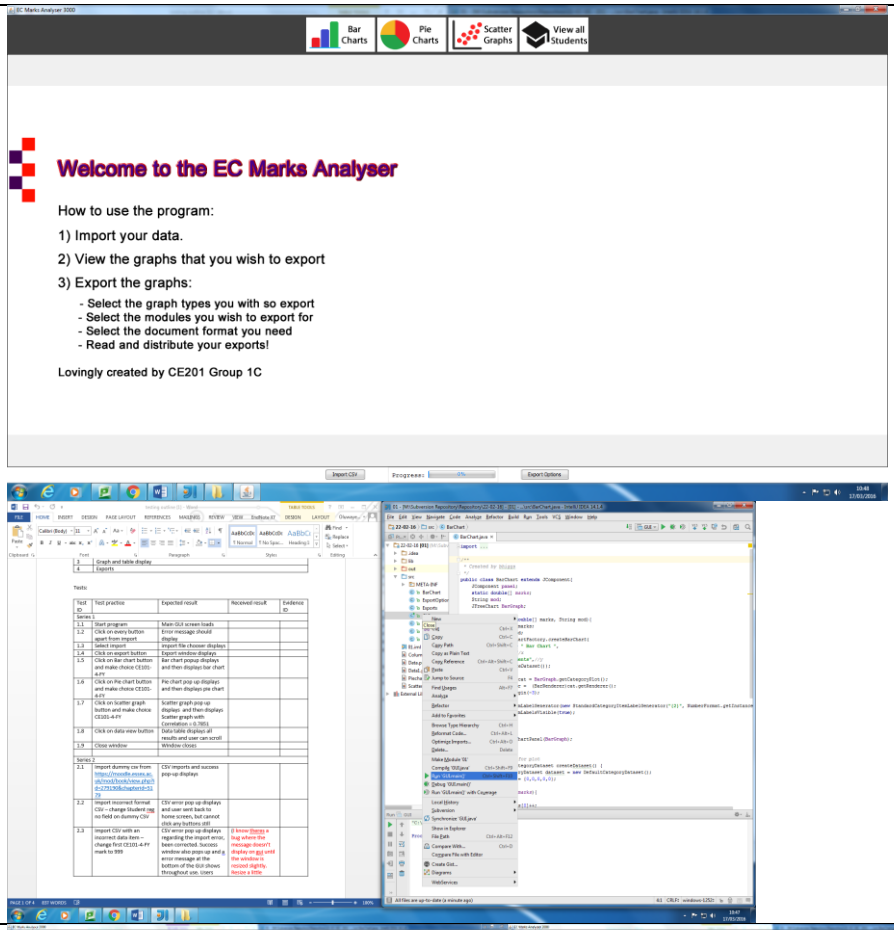
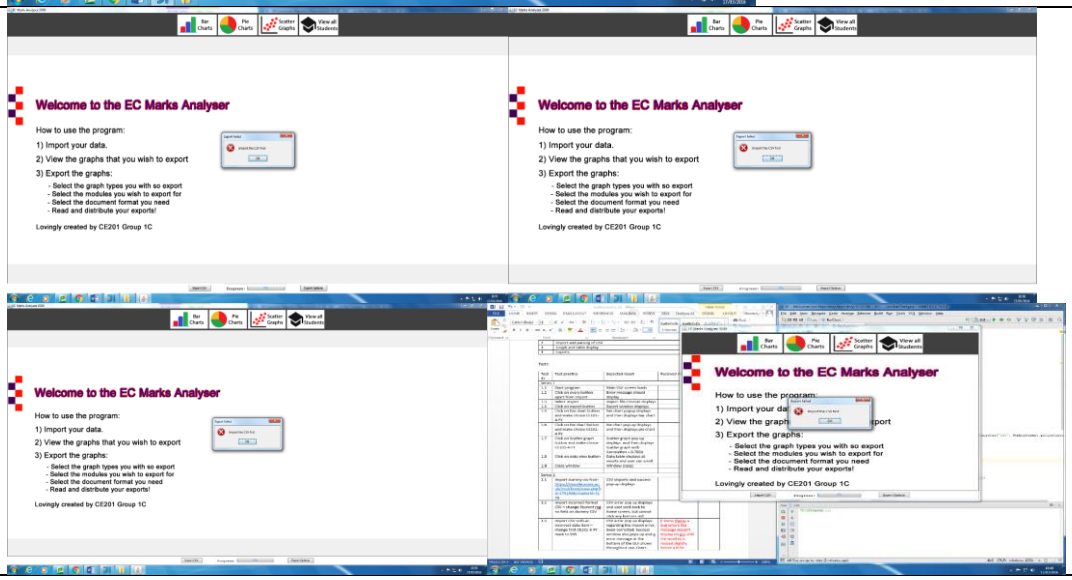


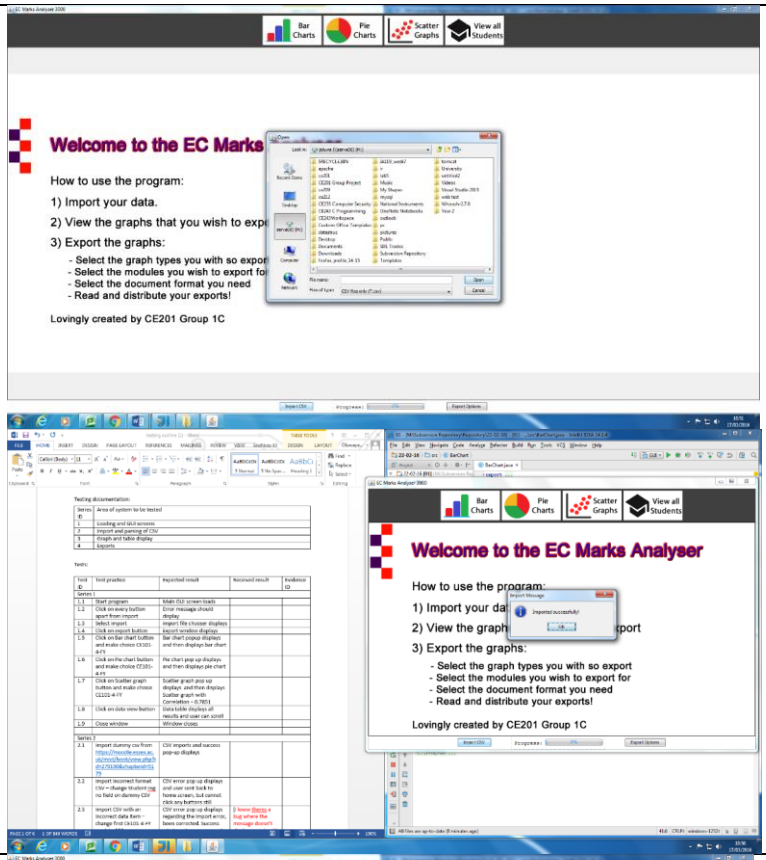
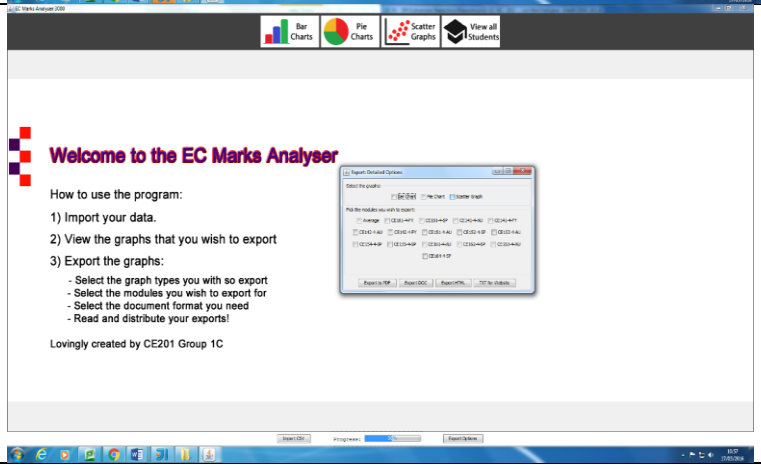
Second Gantt chart



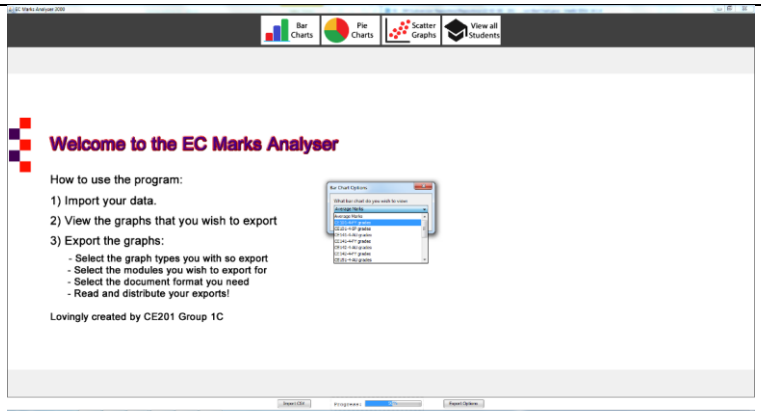
Appendix 3: Testing Evidence

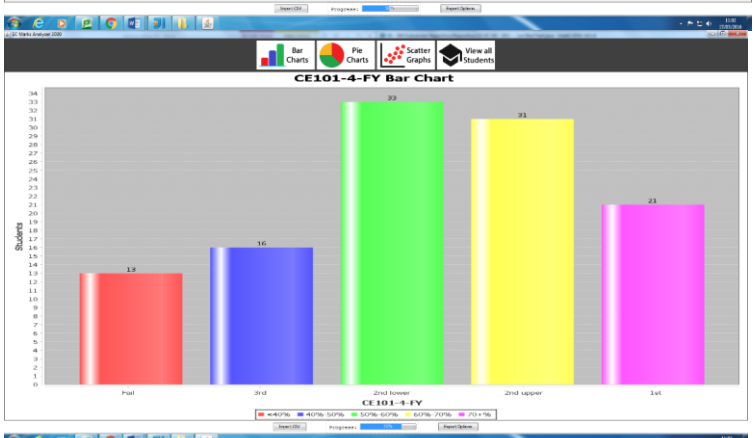
Evidence of result

Evidence ID	Screenshot or sample of export
1	
2	

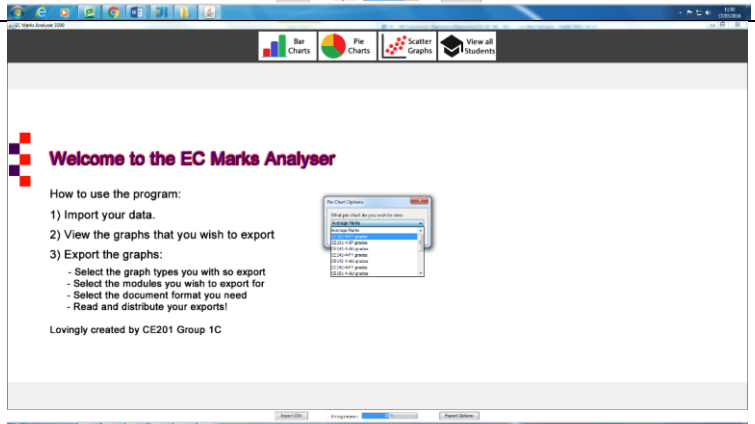
3	 Bar Charts Pie Charts Scatter Graphs View all Students Welcome to the EC Marks How to use the program: 1) Import your data. 2) View the graphs that you wish to exp 3) Export the graphs: - Select the graph types you with so expor - Select the modules you wish to export fo - Select the document format you need - Read and distribute your exports! Lovingly created by CE201 Group 1C Welcome to the EC Marks Analyser How to use the program: 1) Import your da 2) View the graph 3) Export the graphs: - Select the graph types you with so export - Select the modules you wish to export for - Select the document format you need - Read and distribute your exports! Lovingly created by CE201 Group 1C
4	 Bar Charts Pie Charts Scatter Graphs View all Students Welcome to the EC Marks Analyser How to use the program: 1) Import your data. 2) View the graphs that you wish to export 3) Export the graphs: - Select the graph types you with so export - Select the modules you wish to export for - Select the document format you need - Read and distribute your exports! Lovingly created by CE201 Group 1C

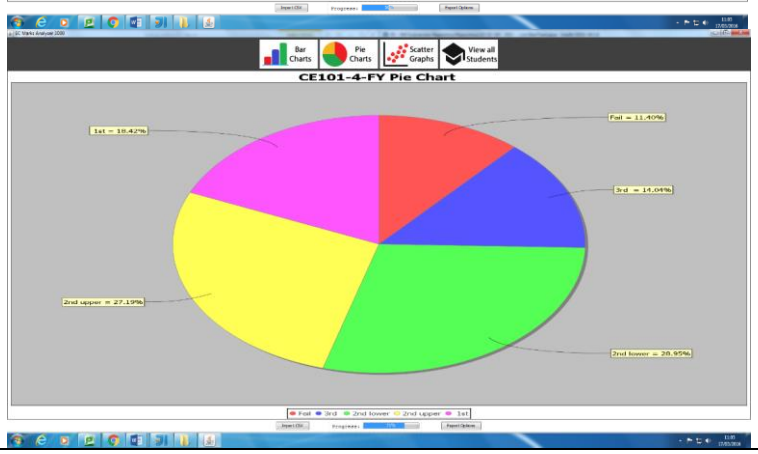
5


Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

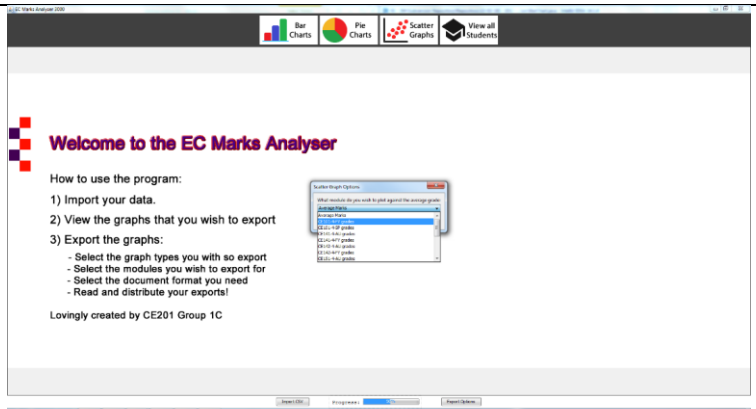

Bar Charts
Pie Charts
Scatter Graphs
View all Students
CE101-4-FY Bar Chart
Students
13
16
33
31
21
Fail
3rd
2nd lower
2nd upper
1st
CE101-4-FY

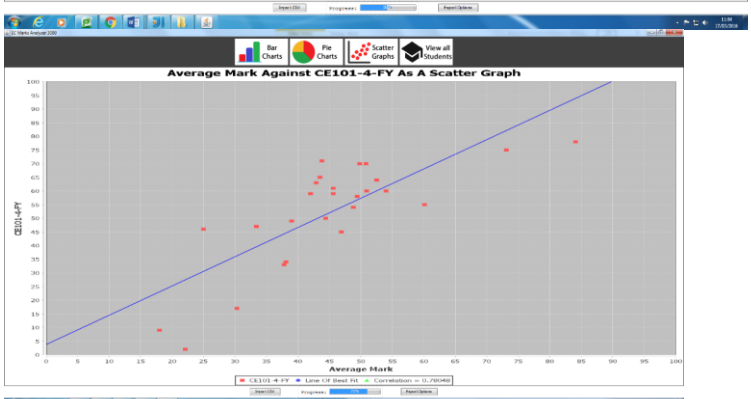
6


Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

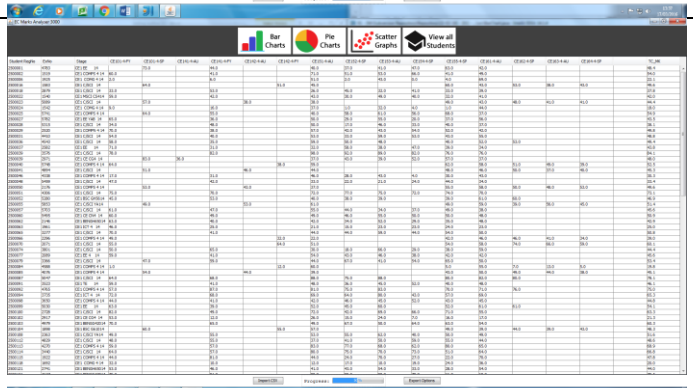

Bar Charts
Pie Charts
Scatter Graphs
View all Students
CE101-4-FY Pie Chart
1st = 18.42%
Fail = 11.40%
3rd = 14.04%
2nd upper = 27.19%
2nd lower = 28.95%

7

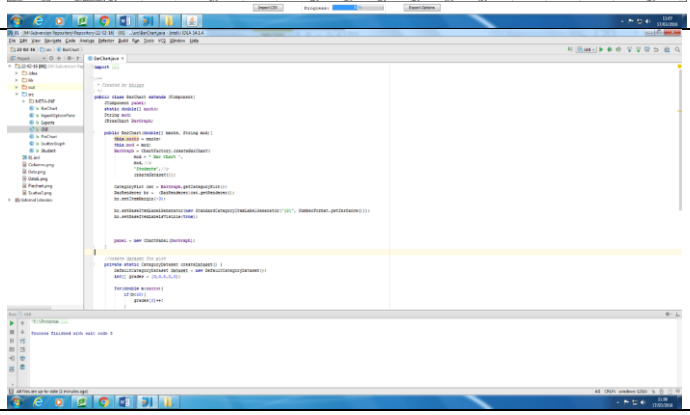

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

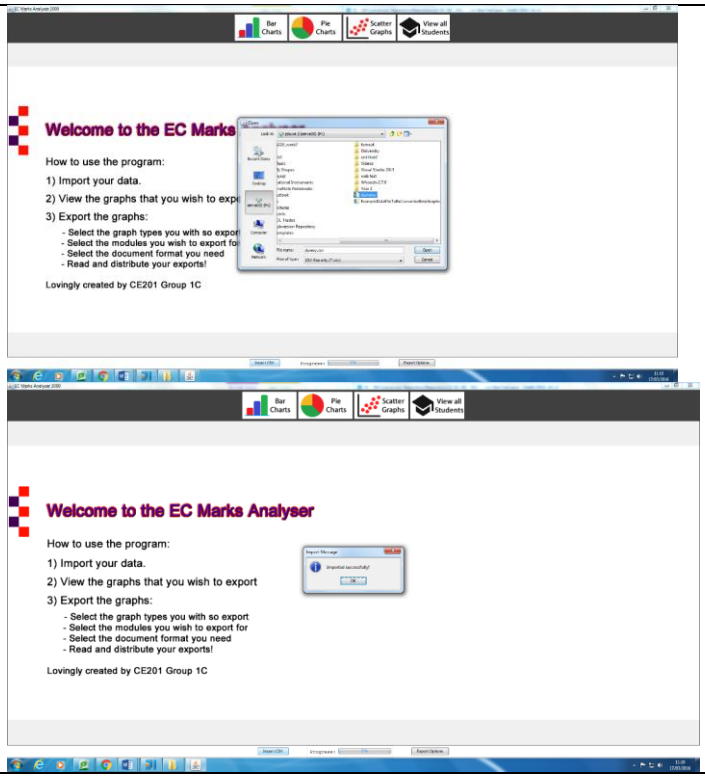

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Average Mark Against CE101-4-FY As A Scatter Graph
CE101-4-FY
Average Mark

8

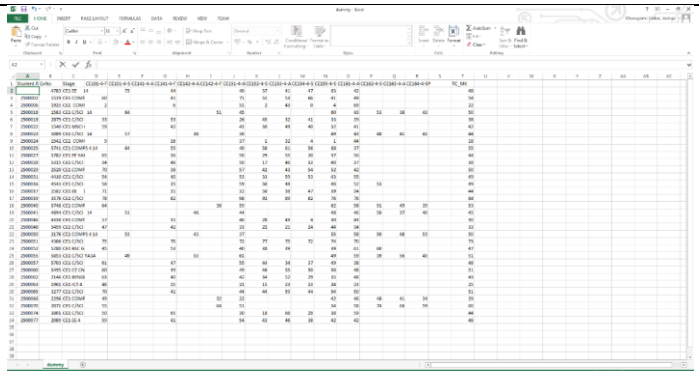

Bar Charts
Pie Charts
Scatter Graphs
View all Students

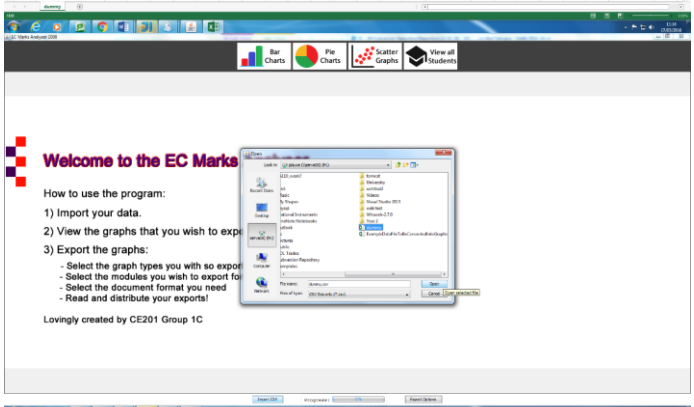
9

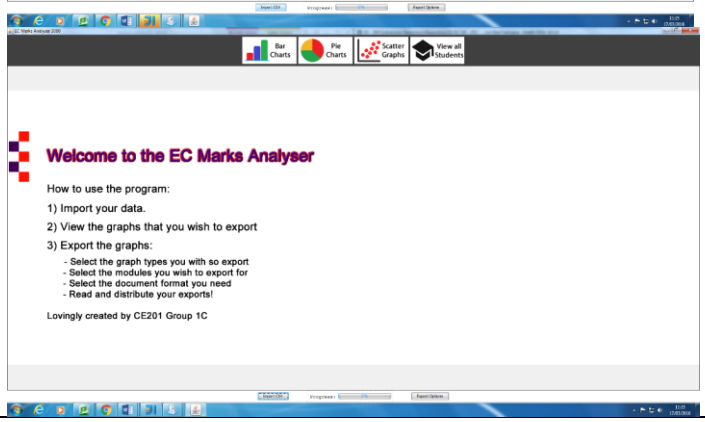


10	 Bar Charts Pie Charts Scatter Graphs View all Students Welcome to the EC Marks How to use the program: 1) Import your data. 2) View the graphs that you wish to exp 3) Export the graphs: - Select the graph types you with so expor - Select the modules you wish to export fo - Select the document format you need - Read and distribute your exports! Lovingly created by CE201 Group 1C Bar Charts Pie Charts Scatter Graphs View all Students Welcome to the EC Marks Analyser How to use the program: 1) Import your data. 2) View the graphs that you wish to export 3) Export the graphs: - Select the graph types you with so export - Select the modules you wish to export for - Select the document format you need - Read and distribute your exports! Lovingly created by CE201 Group 1C

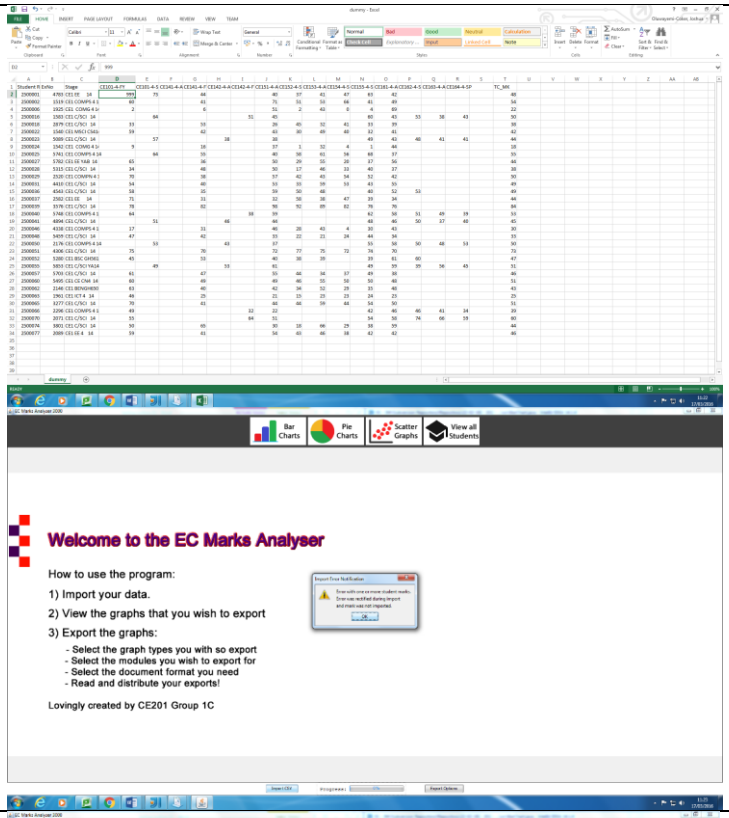
11



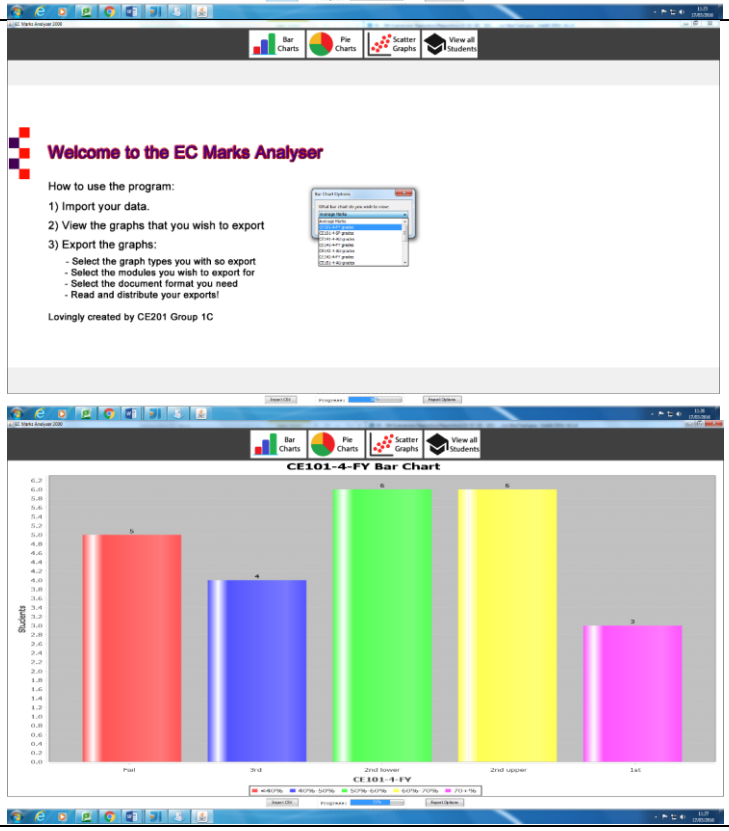

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks
How to use the program:
1) Import your data.
2) View the graphs that you wish to exp
3) Export the graphs:
- Select the graph types you with so expor
- Select the modules you wish to export fo
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C


Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

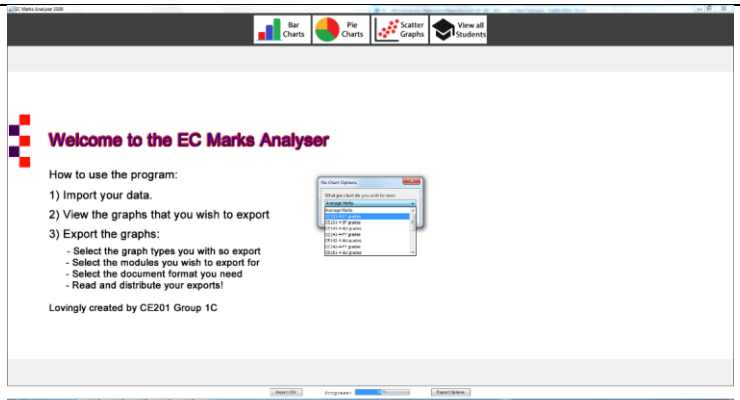
12

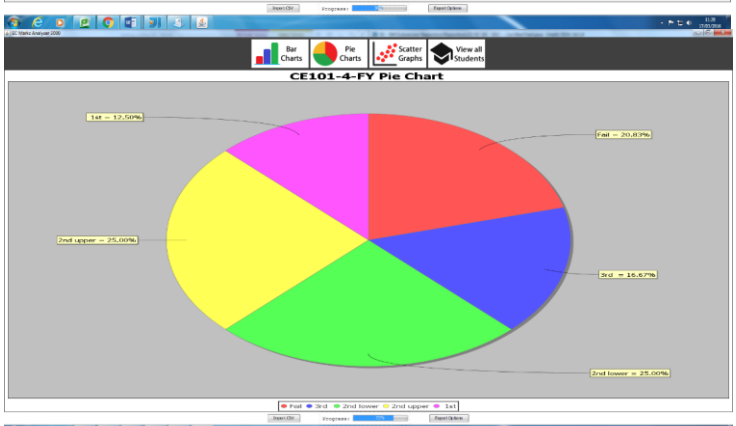

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

13

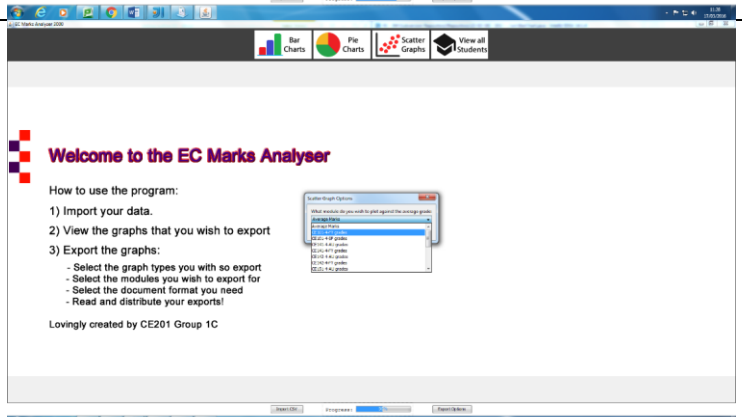

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C
CE101-4-FY Bar Chart
Students
Fail
3rd
2nd lower
2nd upper
1st
CE101-4-FY

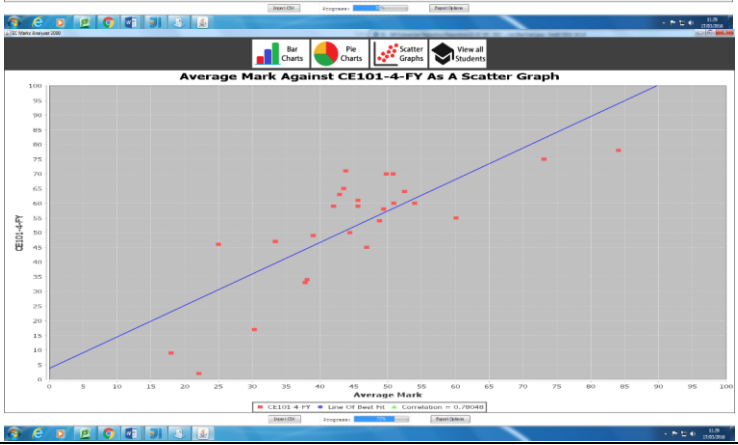
14


Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

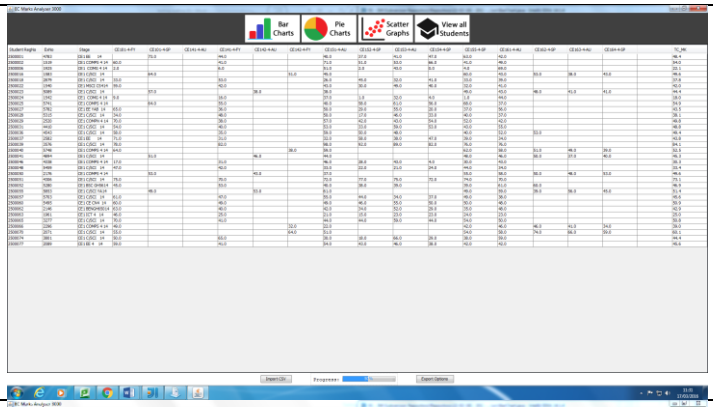

Bar Charts
Pie Charts
Scatter Graphs
View all Students
CE101-4-FY Pie Chart
1st = 12.50%
Fail = 20.83%
2nd upper = 25.00%
3rd = 16.67%
2nd lower = 25.00%
Fail
3rd
2nd lower
2nd upper
1st

15

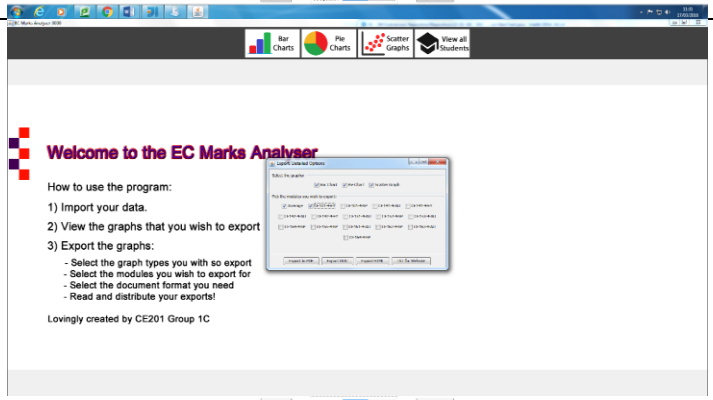

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

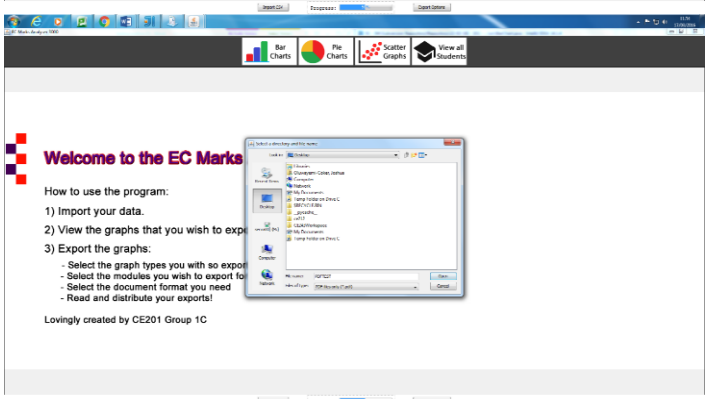

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Average Mark Against CE101-4-FY As A Scatter Graph
CE101-4-FY
Average Mark
CE101-4-FY
Line Of Best Fit
Correlation = 0.78048

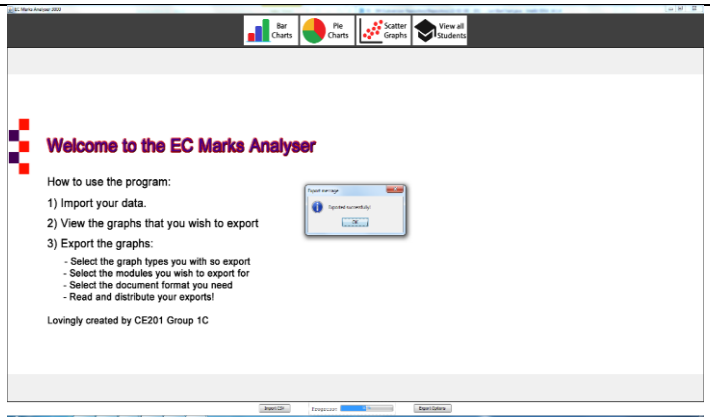
16

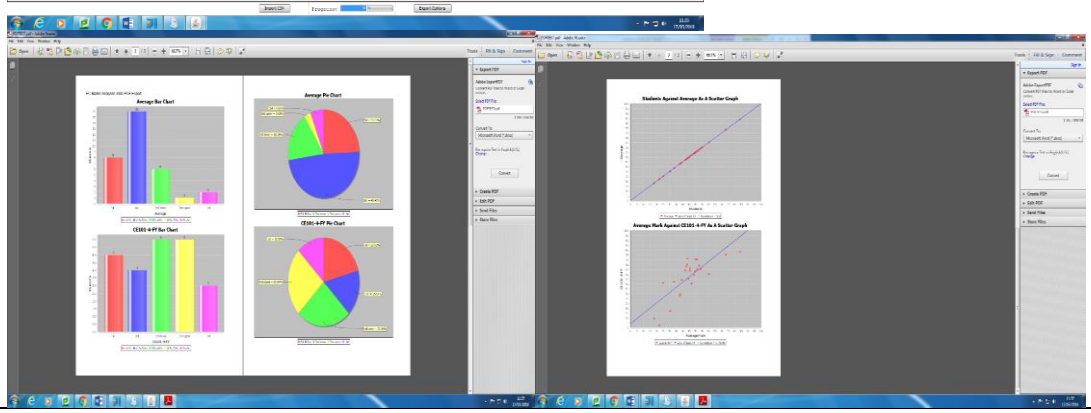

Bar Charts
Pie Charts
Scatter Graphs
View all Students

17

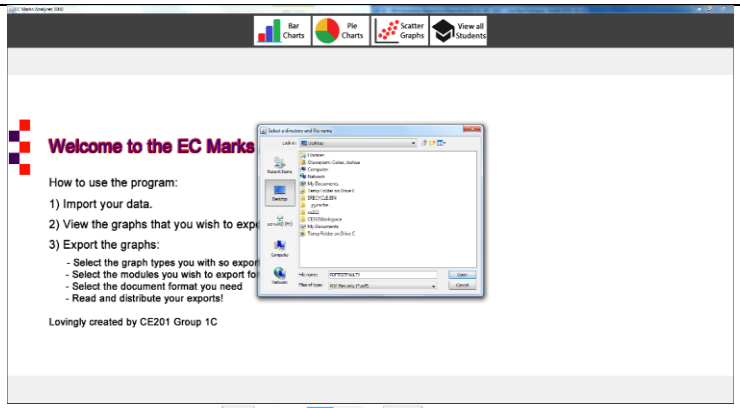

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

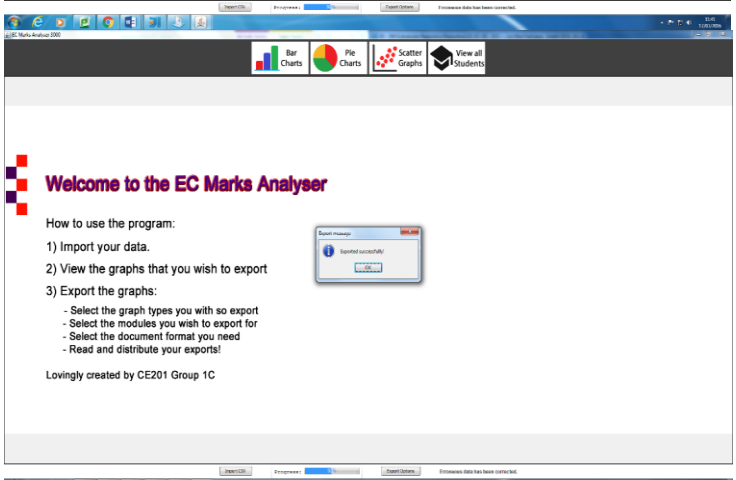

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks
How to use the program:
1) Import your data.
2) View the graphs that you wish to exp
3) Export the graphs:
- Select the graph types you with so expor
- Select the modules you wish to export fo
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

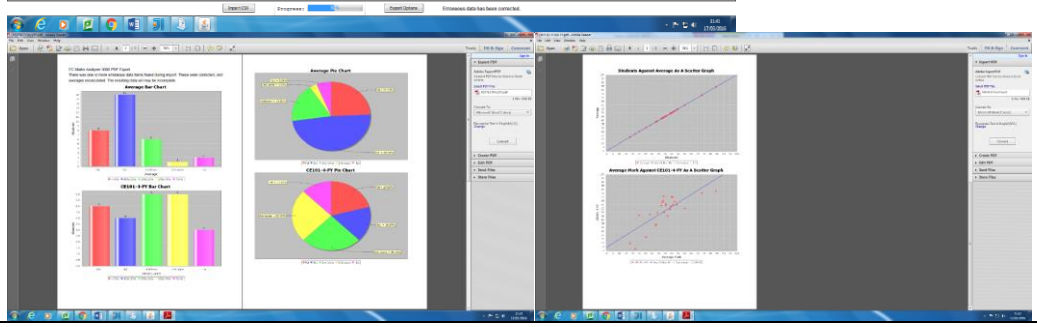

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C



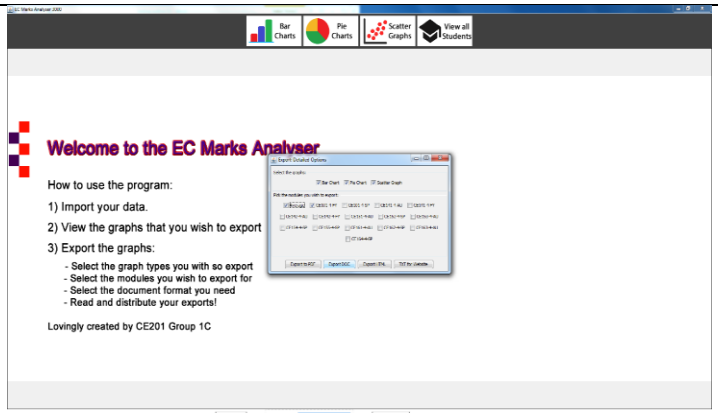
18

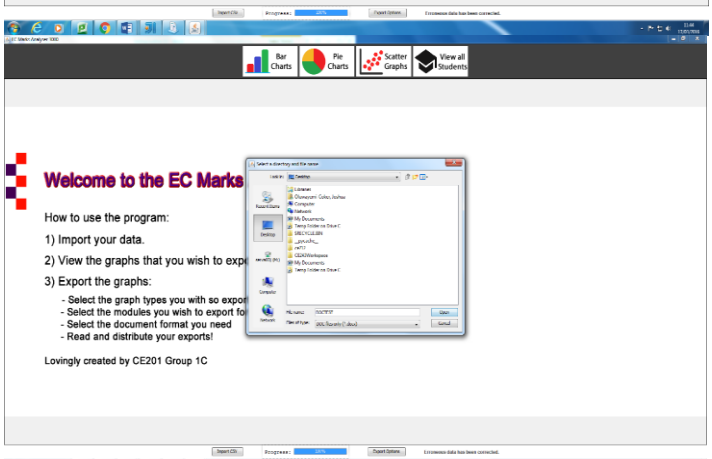

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks
How to use the program:
1) Import your data.
2) View the graphs that you wish to exp
3) Export the graphs:
- Select the graph types you with so expor
- Select the modules you wish to export fo
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

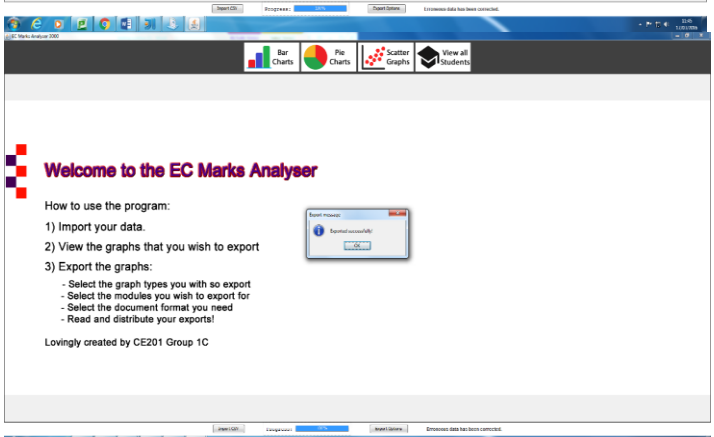

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

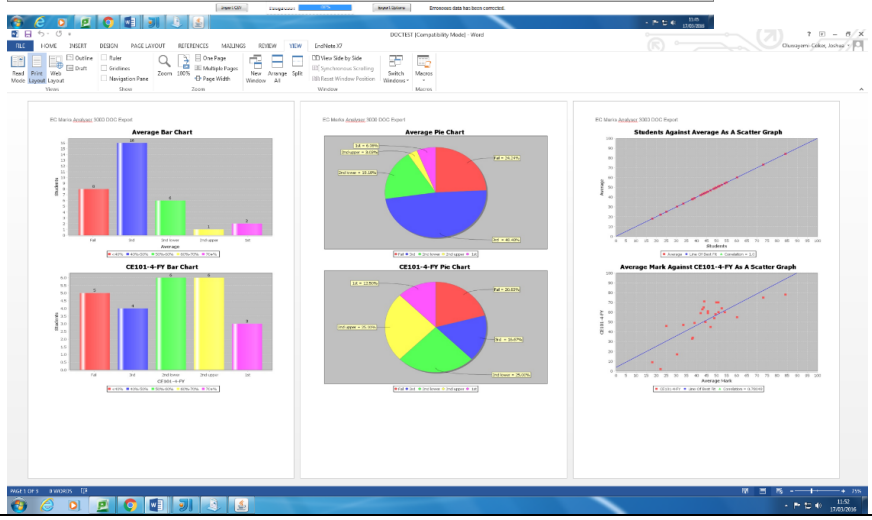


19

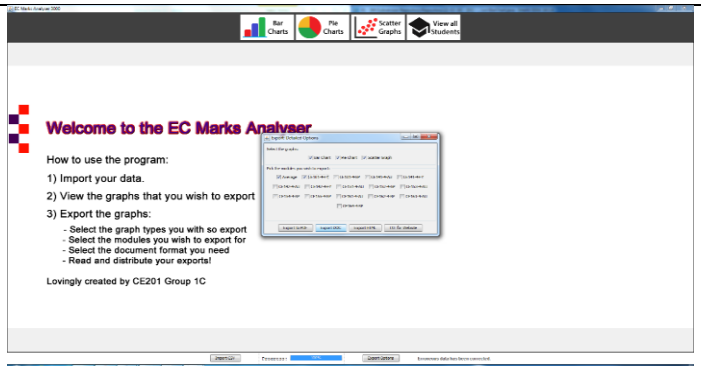

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

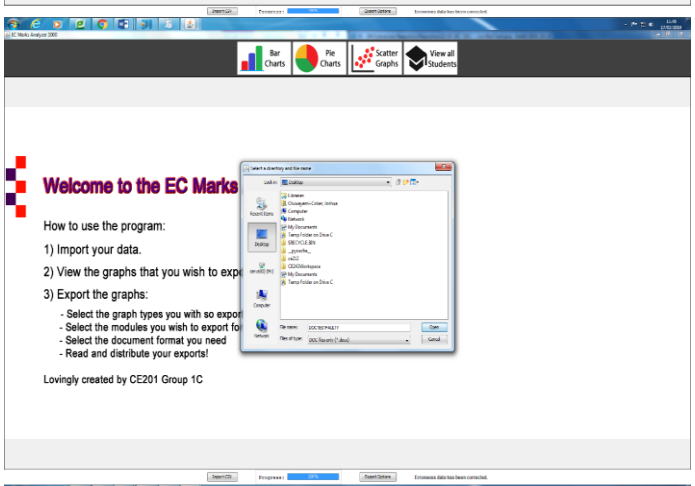

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks
How to use the program:
1) Import your data.
2) View the graphs that you wish to exp
3) Export the graphs:
- Select the graph types you with so expor
- Select the modules you wish to export fo
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

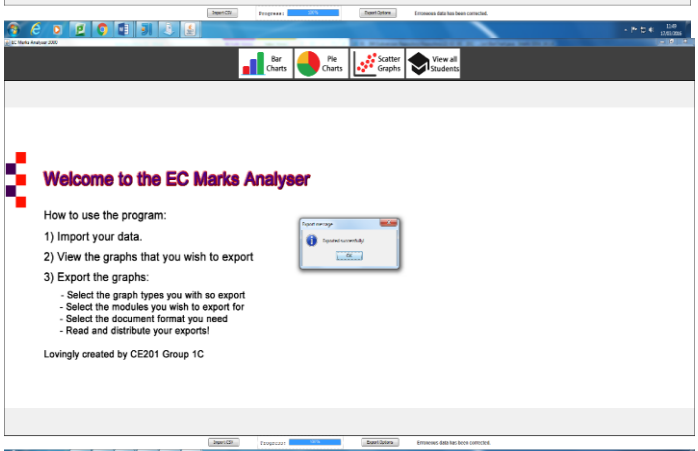

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

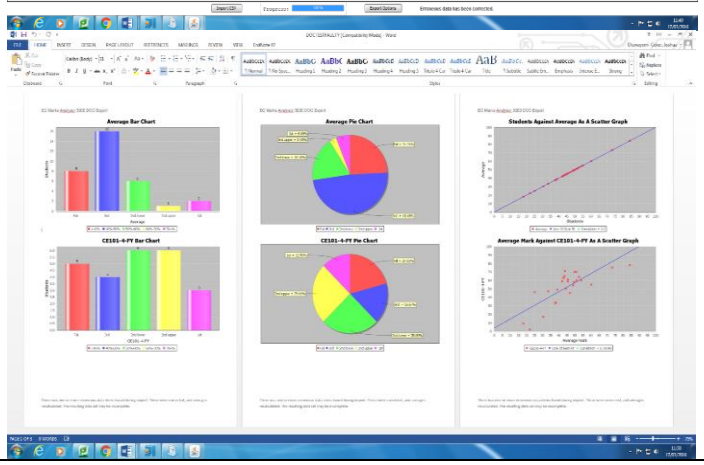

Average Bar Chart
Average Pie Chart
Students Against Average As A Scatter Graph
CE101-4-PY Bar Chart
CE101-4-PY Pie Chart
Average Mark Against CE101-4-PY As A Scatter Graph

20

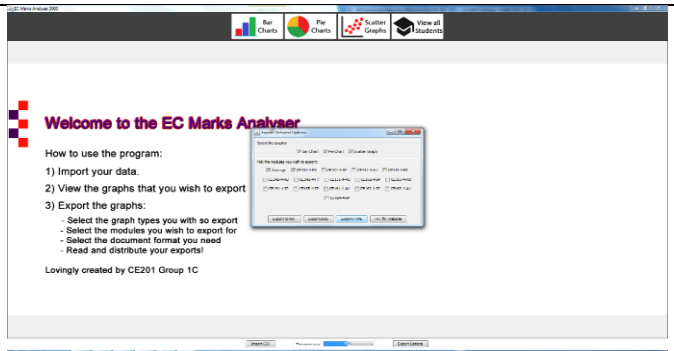

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

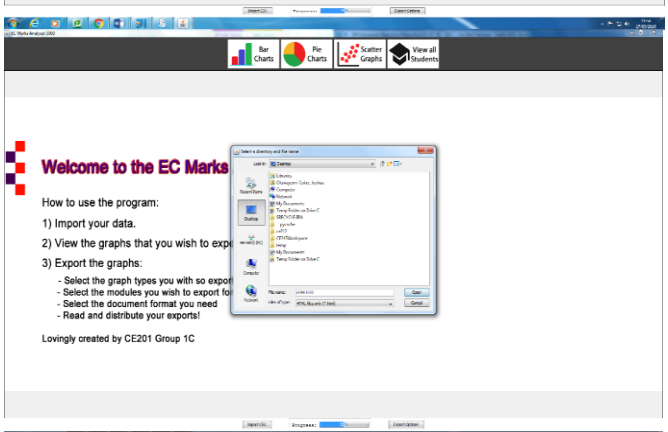

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks
How to use the program:
1) Import your data.
2) View the graphs that you wish to exp
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

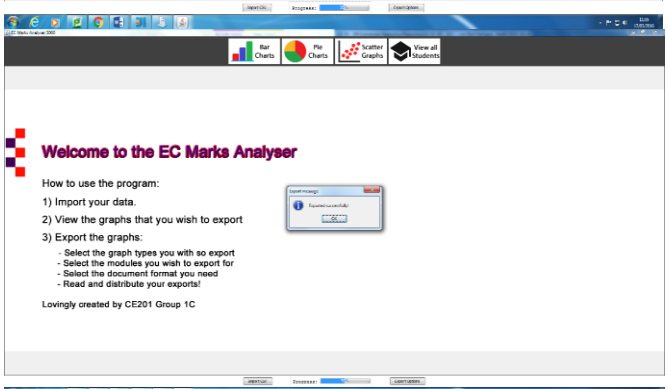

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

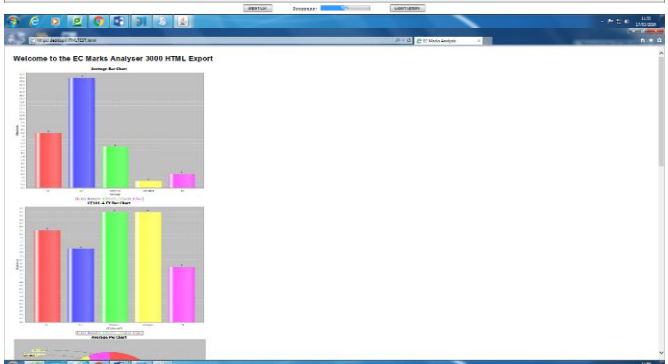

Average Bar Chart
Average Pie Chart
Students Against Average As A Scatter Graph
CE101-4-PY Bar Chart
CE101-4-PY Pie Chart
Average Mark Against CE101-4-PY As A Scatter Graph

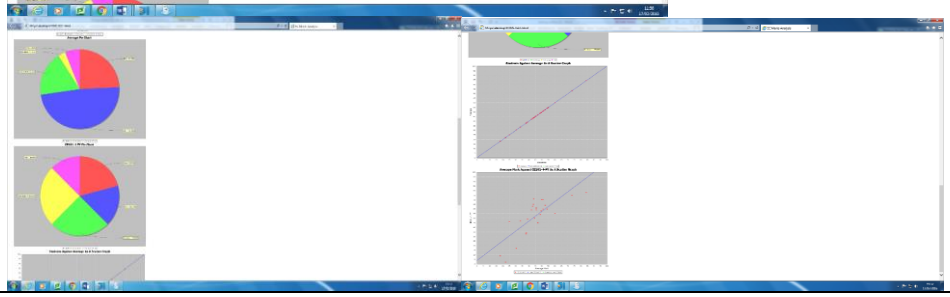
21


Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

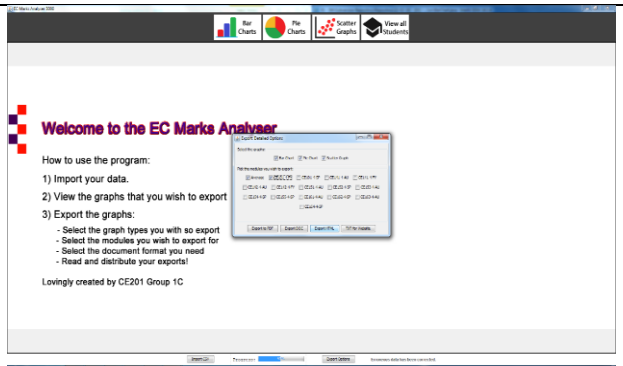

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks
How to use the program:
1) Import your data.
2) View the graphs that you wish to expo
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

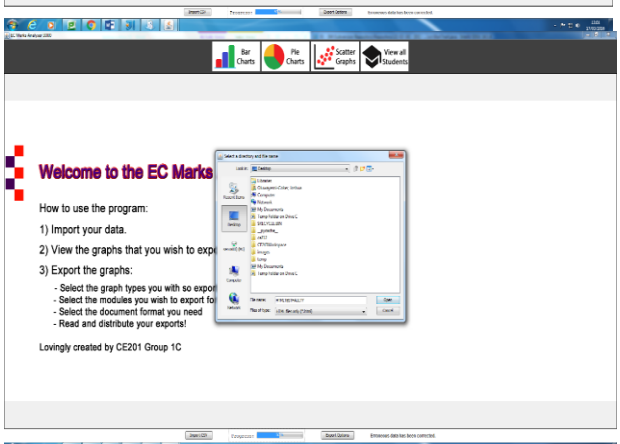

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

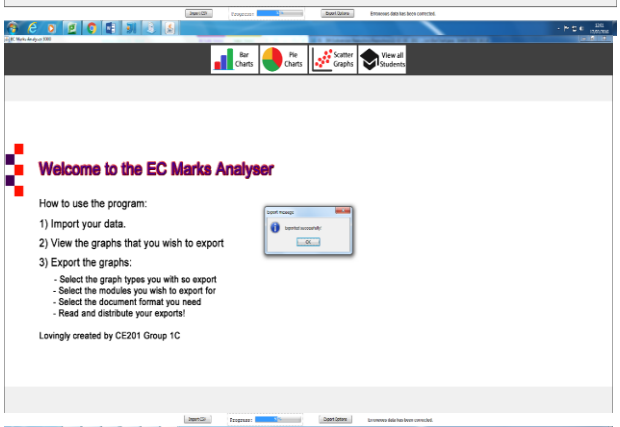

Welcome to the EC Marks Analyser 3000 HTML Export

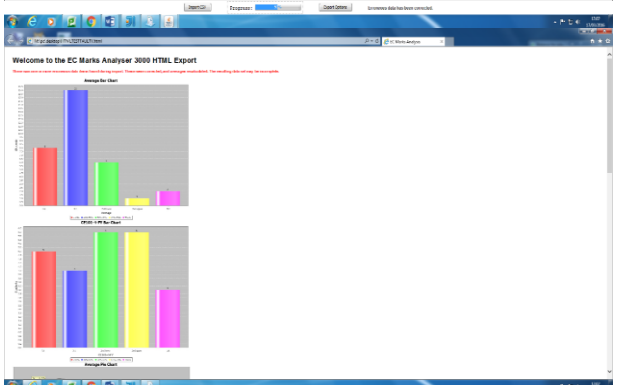


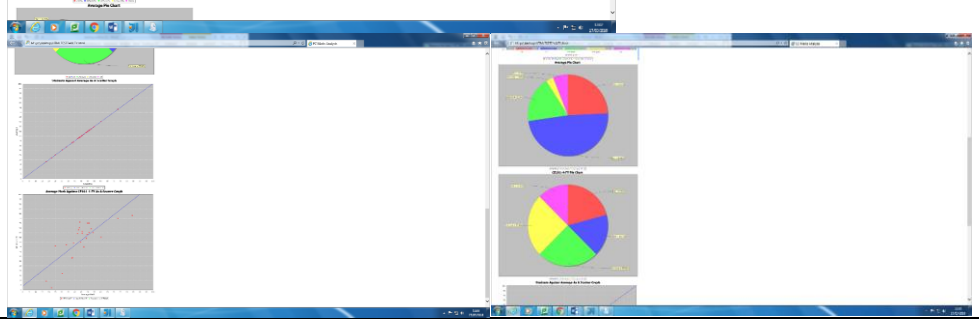
22


Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

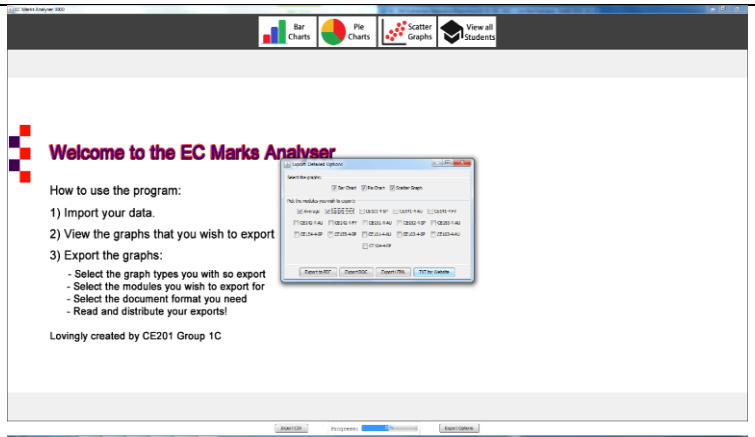

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks
How to use the program:
1) Import your data.
2) View the graphs that you wish to expo
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export fo
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

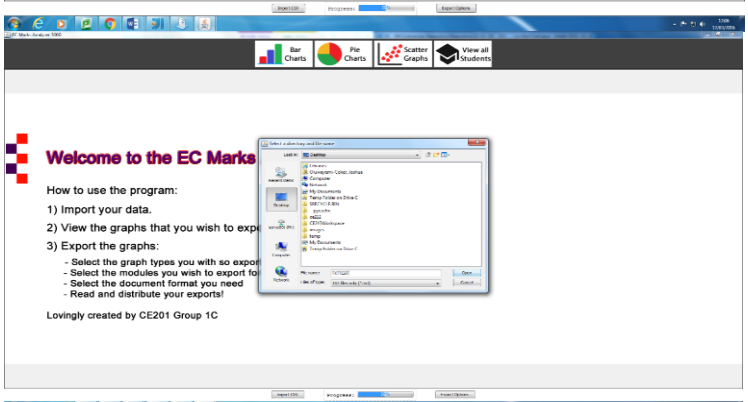

Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C

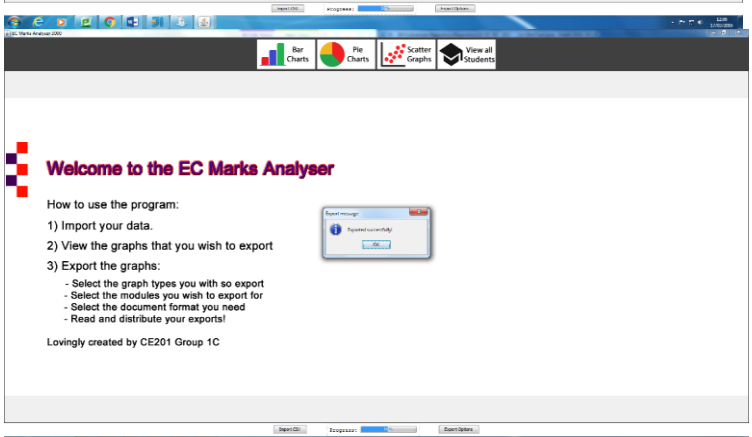

Welcome to the EC Marks Analyser 3000 HTML Export



23


Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C


Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks
How to use the program:
1) Import your data.
2) View the graphs that you wish to expo
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C


Bar Charts
Pie Charts
Scatter Graphs
View all Students
Welcome to the EC Marks Analyser
How to use the program:
1) Import your data.
2) View the graphs that you wish to export
3) Export the graphs:
- Select the graph types you with so export
- Select the modules you wish to export for
- Select the document format you need
- Read and distribute your exports!
Lovingly created by CE201 Group 1C