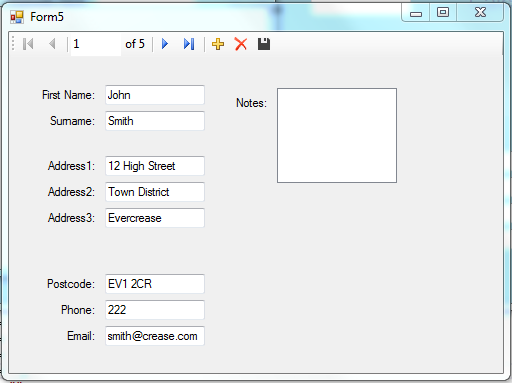
**P1 – Explain the key features of event driven programs**

**Introduction**

In this report, I will be explaining the key features of event driven programs. I will conclude with why triggers and timing is important.

EDP is service-oriented

Service-oriented are programs that provide services to the user. These services could be run over a network. This is often appeared in the background. Therefore, it has no user interface to use but it has the ability to be stopped and started again. An example of a service-oriented is grabbing database and locating the information into the program. Referring to Figure 1.1, it shows how it works. Once the button is clicked, the whole process that runs in the background; it is the service provided. The database works by grabbing the information from Microsoft Access and putting it into the Textboxes shown above.

Figure 1.1

EDP is time-driven

This is the time-driven feature in EDP that runs on a time-based. An example for this could be an hour a day. This is pre-set by the user on a time trigger for this to be complete. An example could be that an anti-virus is scheduled once a week to scan the computer. As service-oriented, it can be stopped too.

Event handlers

Event handlers are a section of codes that indicates an event triggers. They are many of them. It can affect the whole program depending on what the section is all about. For example, if I were to put in a code that closes the program, I would need to program it within the button for the trigger to be set. Referring to Figure 1.2, it shows an example of an event handler. The code below closes the form. This is the instruction of the event.



Figure 1.2

Trigger functions

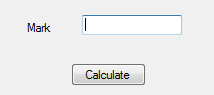
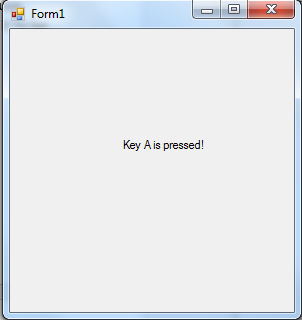
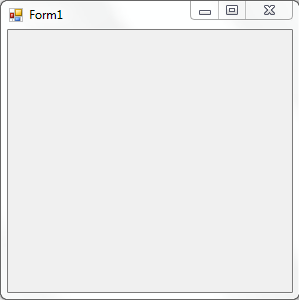
Trigger functions are used in EDP for it to respond and triggered by the user. Each object used in VB.Net is different and it would not be the same. The trigger functions is important, because some of the triggers are flexible and you can use them together. For example, only a button is used for it to be clicked on. For one object, it has one trigger function. Referring to Figure 1.3, it shows an example of a trigger function. Once I have button has been pressed, it calculates the mark for the user. The job is for it to be triggered and respond.

Figure 1.3

Events e.g. mouse, keyboard, form

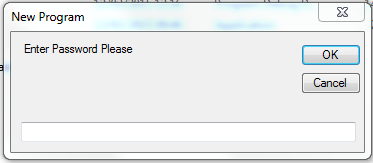
Events is subjected into categorises. One of them is mouse events. This refers to the actions that is used on the mouse. For example, if the program was to move the mouse to the left, it will recognise it at the same time. Another is keyboard events. This is the same. Program was to press a key on the keyboard, it will recognise it and do the event. For example, CTRL, ALT and DELETE allows the user to lock the system and more. The last one that I am going to talk about is Form. Forms are used to provide applications for users.

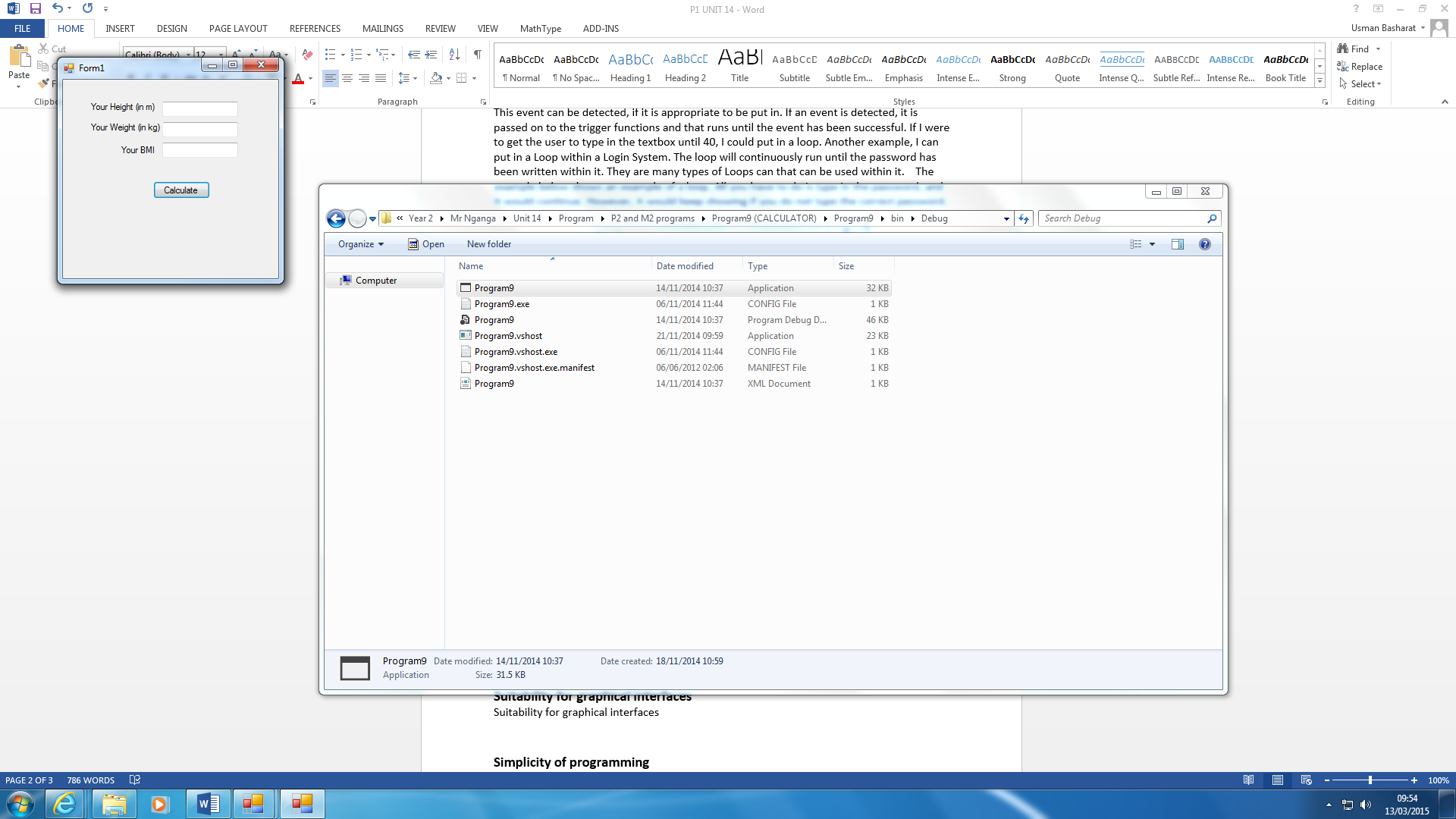


Keyboard Event

Event loops

This event can be detected, if it is appropriate to be put in. If an event is detected, it is passed on to the trigger functions and that runs until the event has been successful. If I were to get the user to type in the textbox until 40, I could put in a loop. Another example, I can put in a Loop within a Login System. The loop will continuously run until the password has been written within it. They are many types of Loops can that can be used within it. The example below shows an example of a loop. All you have to do is type in the password, and it would continue. However, it would keep showing if you do not type the correct password.



Flexibility

One of the key of event driven programming is its flexibility. The range of events that can take place at once is huge. The BMI calculator shows that it has more than one event within the form. You can type in your height, weight and it would give your BMI by clicking the calculate button.

Suitability for graphical interfaces

GUI is a clear example of how it allows users to interact with the program whilst programming it. This makes is very easy to use. Moreover, if they know the program language, it is even more effective. Some features that might include within a GUI could be:

* Pointer

Pointer is a small symbol that displays on the screen to move objects around.

* Icons

Icons is small pictures that is represented as commands. Therefore, the user uses the pointer with the icon to execute a command.

* Menus

Menu bar gives you a choice full of commands. For example, HOME – SAVE AS; this would save the program.

* Desktop

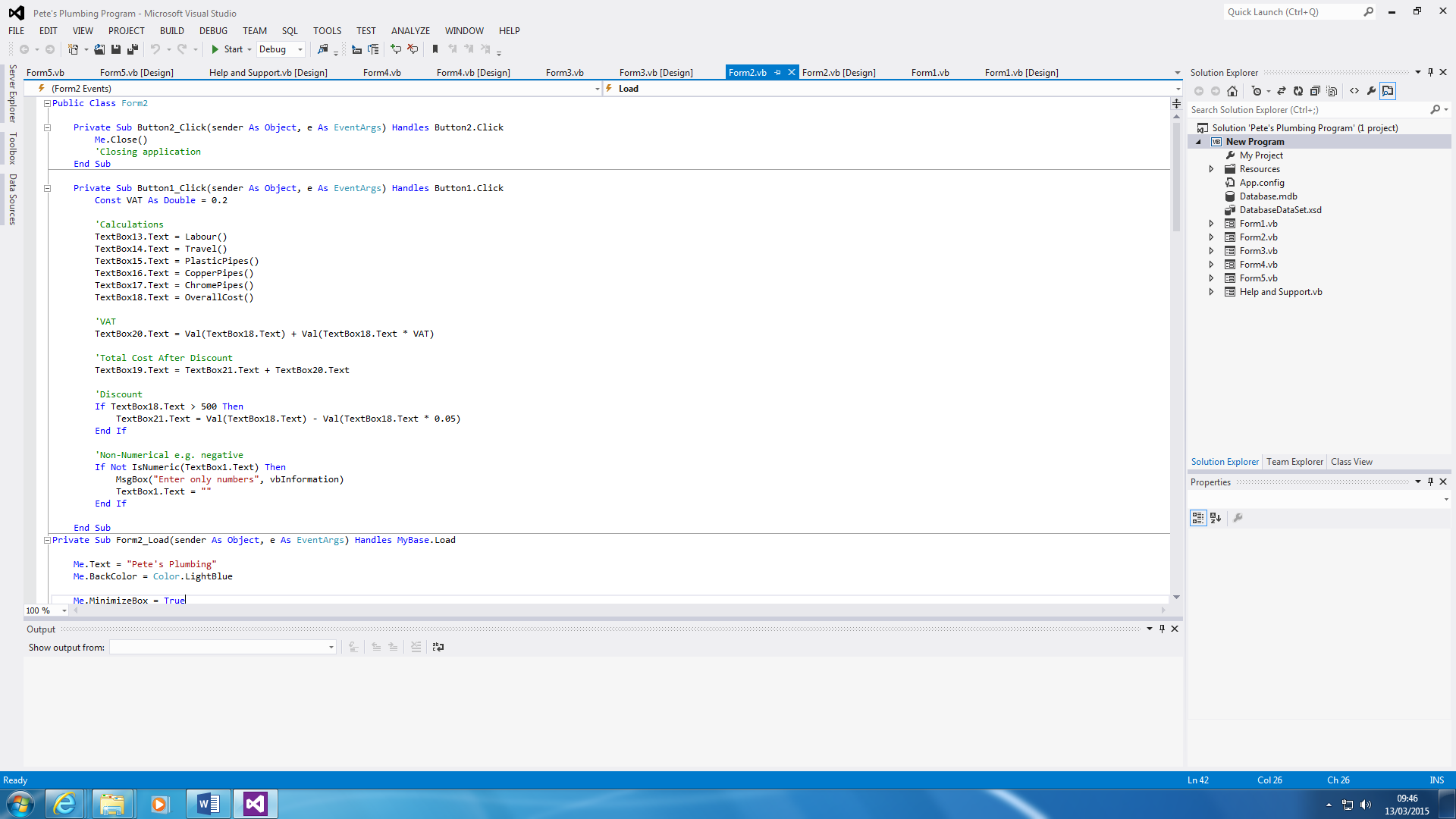
This is what is on the home page. This is where you open up the program.

Simplicity of programming

Simplicity of programming is the visual aspect of programming makes programming much easier for event-driven programming. It makes life easier by the user using another code to put into the form. Making it simple and easy would save a lot of time for the user to make; even a calculator would be done within a lesson.

Ease of development

Ease of development is making it easier for the user. What the user needs to do is type in a code for a simple event and it will run. For example, close the button. Testing the event makes it easier too.



Conclusion

To conclude, I think that triggers and timing are important because I feel that both of these equal to the event being a success. Both of them work together by the trigger being released by the time. For example, if I were to click on a button, it would need time for it to be executed. Once I have triggered it, the timing of it being complete would equal to both of them being important.

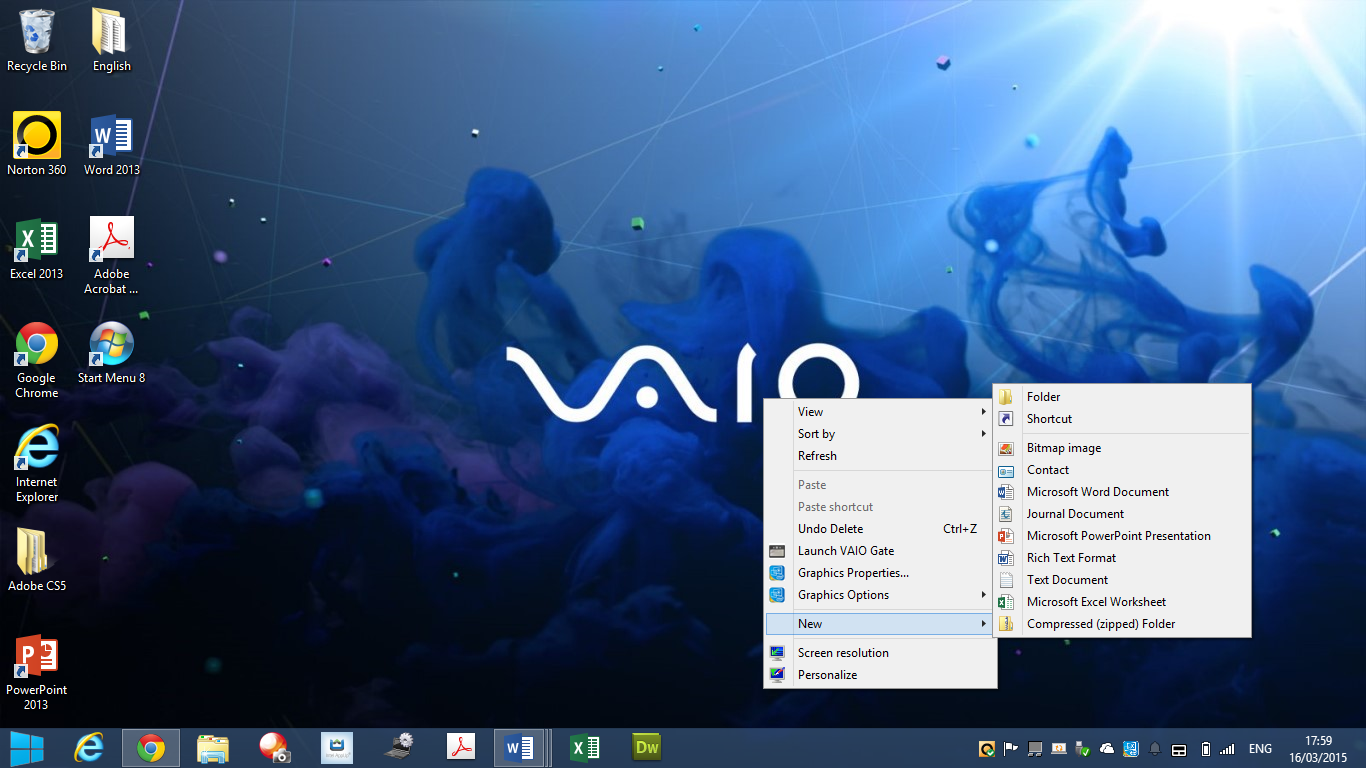
P1 - Reference

<http://reece-burrows-media.blogspot.co.uk/2013/11/key-features-of-event-driven-programming.html>

**M1 – Discuss how an operating system can be viewed as an event driven application**

**Introduction**

In this report, I will be discussing how Windows can be operated as an event-driven application. For evidence, I will display five screenshots with annotation to explain the action, trigger and event for each screenshot.



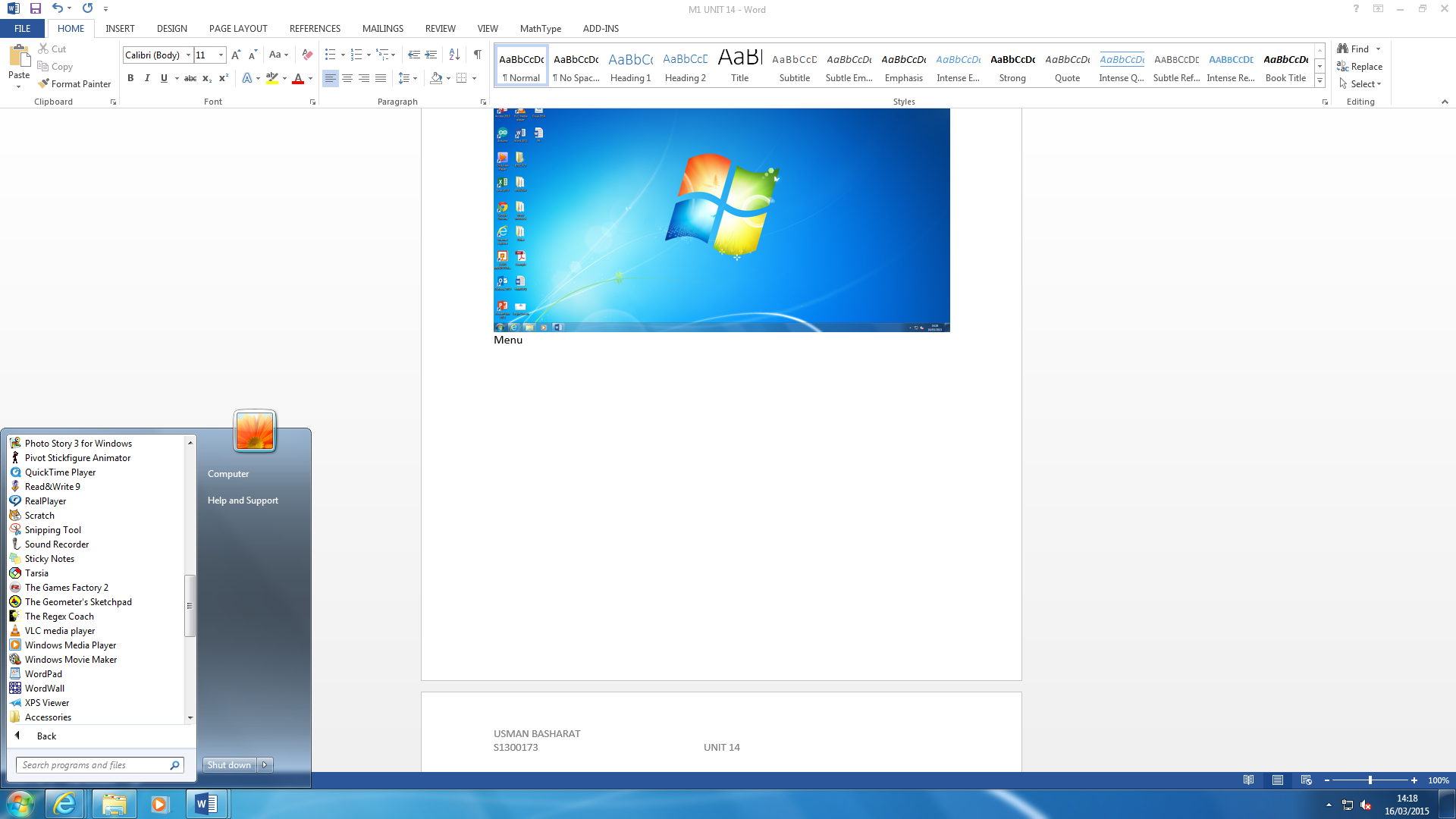
**Screenshots**

Desktop



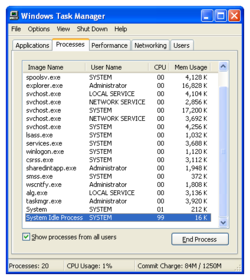
This is a desktop shown on Windows 8. The event that is taken place within this event-driven application is for the folder to be created on the desktop. To do this, the action is to use the mouse by clicking right-click. The trigger is, once you go on NEW 🡪 FOLDER, you click on it. The event is taken place by creating a folder.

**Menu**

****

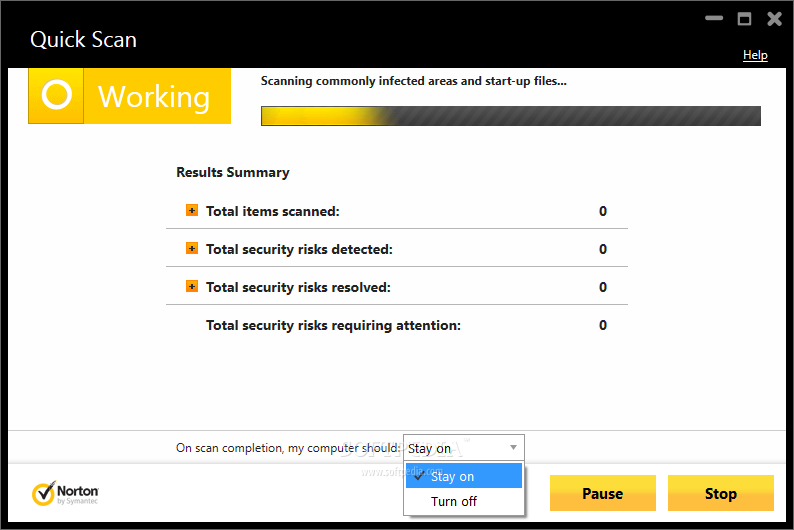
This is a start button shown on Windows 7 operating system. The action that is taking place within this event driven application is for the user to click on the button. This triggers the start-up menu to be shown. The user uses the mouse for the event to take place. The event that is to take place is for the user to open up the application needed. For example, if I want the calculator application to open up, I would click on the start button – triggers the menu to open up – find the calculator and click on it. This is the event by the user finding the application and clicking it.

**Task Manager**

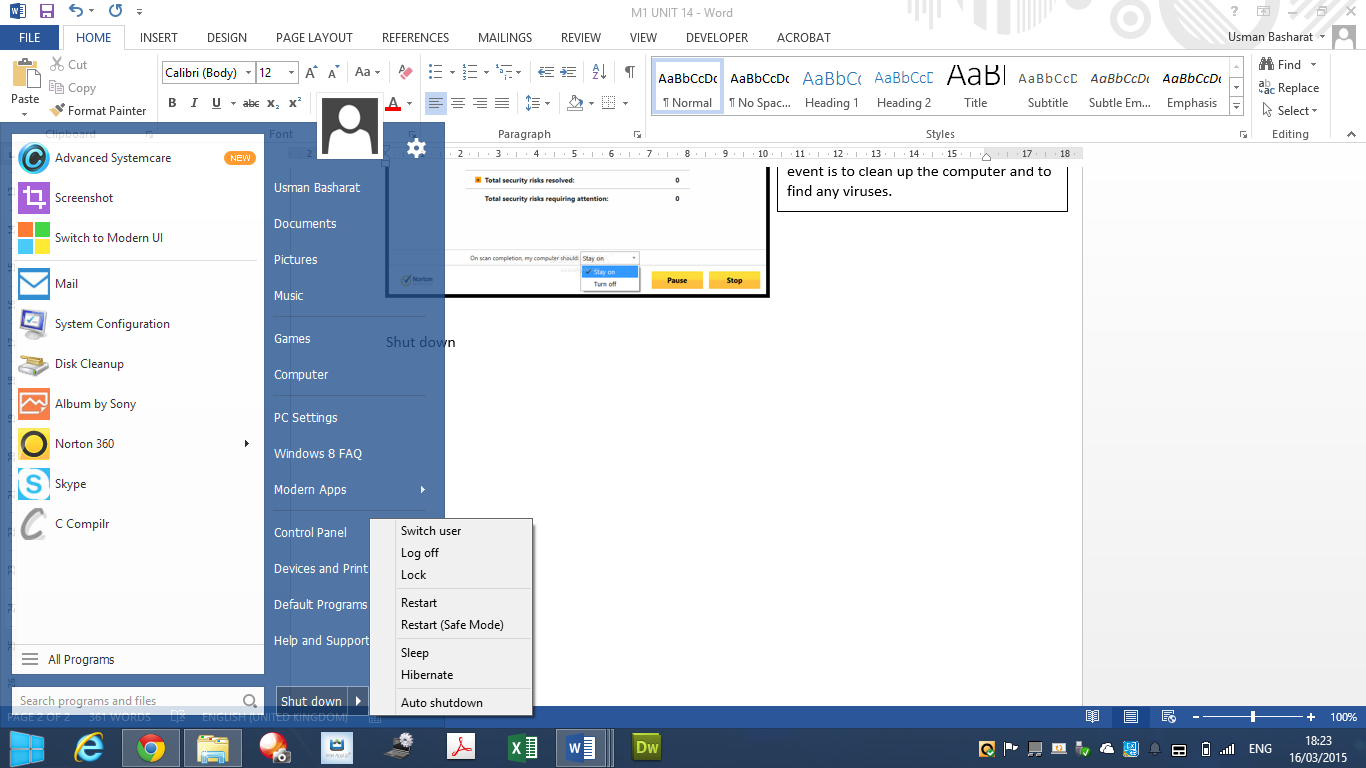


Task Manager is found by the action first taking place. This time, the keyboard is in place. The user needs to click ALT CRTL + DELETE all together to browse up the options. Once this is done, the trigger is for the user to use the mouse to click on ‘TASK MANAGER’ and to sense that it has been clicked on. The event is for the user to end any applications that have been crashed. For example, if Microsoft Word were crashed, I would go on Task Manager and end the process as shown on the image.

**Anti-Virus**



In this case, you use the mouse again to find the application. The action is to double-tap on the application. By clicking scan, it triggers the event. It should sense it. The event is to clean up the computer and to find any viruses.

**Shut down**

As you can see in the screenshot, there are many options to shut the computer down. The action is to simply shut the computer down. To do this, you click on the start button and choose which one you want to do. The trigger is for the computer to sense that it has been clicked on for the event to take place.

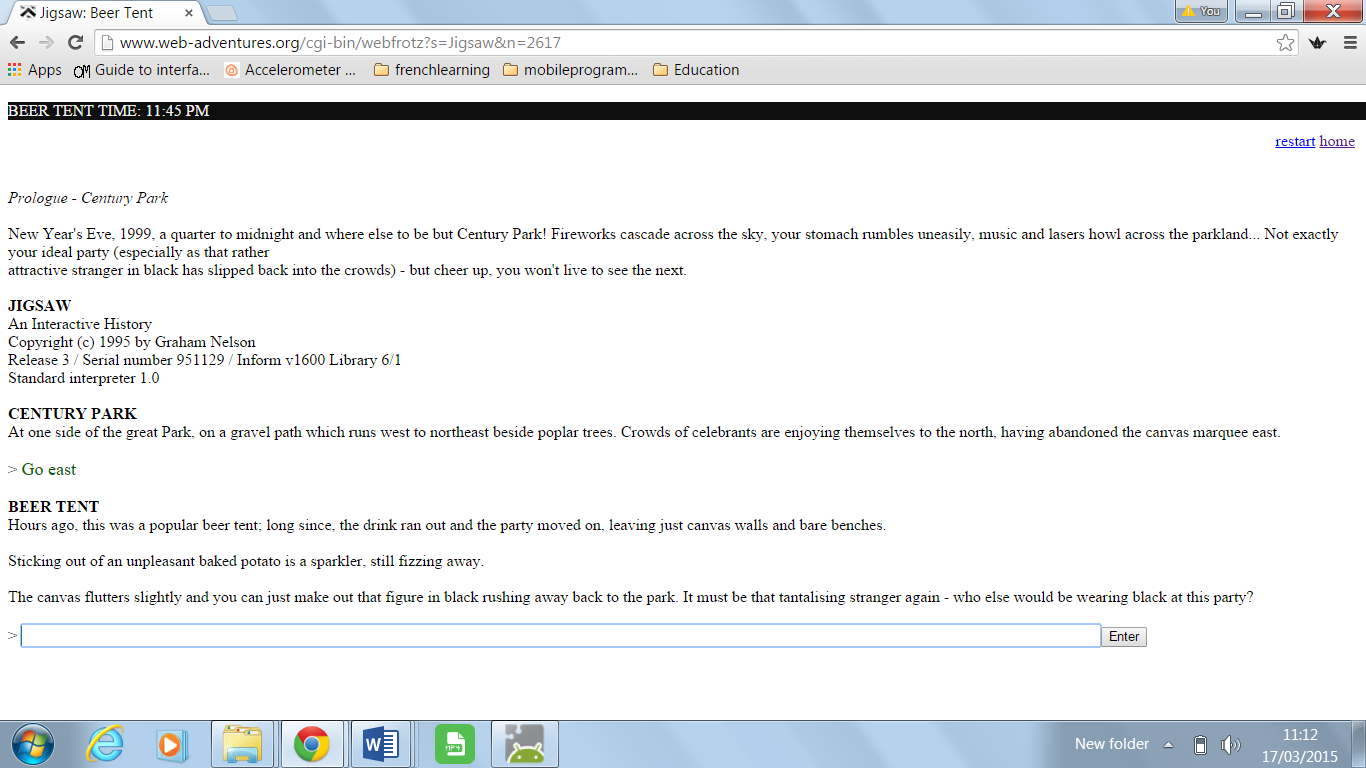
Some of the events shown above that have been taken place are EDP, because they use mouse event or keyboard event.

**D1 – Evaluate how suitable an event-driven programming languages is for non-graphical applications**

**Introduction**

In this report, I will be evaluating how suitable event-driven programming language is for non-graphical applications.

**Jigsaw**

Event-driven means that the program that takes an event and handles it. In a graphical user interface, it uses keystroke, mouse movement, click, touch etc. I feel that event-driven programming can be suitable for non-graphical applications, because all you need to use is the keyboard which could be complete. For example, JIGSAW game. This is a game where you can only type in. It shows no sign of any graphics. The picture below shows how the game works. You have to type in what you have to do in order to play the non-graphical application. As you can see below, I typed in ‘Go East’, you can see a response below.

For this game to be suitable, three criterions needs to be filled.

* Action

The action is to use read the story at the beginning to understand what is going on. The action is to use the keyboard by pressing ‘Go East’ to get a reply.

* Trigger

The trigger is to press enter and for the program to recognise that you have typed something in as shown above.

* Event

The event is for the user to keep typing and eventually complete the game.

The three features that you need in this game is to be able to press event, keyboard event. These similar events are used in event-driven programming. A non-graphical application does not need to use a mouse. All it needs to use is a keyboard for it to be a non-graphical application, which includes there being no graphical content. As you can see, those requirements have been met. The mouse is not being used with this game. All you do is type for this game.

Non-graphical applications were set out in the 19th century. Back in those days, a mouse was not used. Only a keyboard was used to play games. To answer the question that was set, I feel that event-driven programming language is suitable for non-graphical applications. Event-driven programming language responds to a request made. The request made by this game is pressing a key. This can be related to this game. Jigsaw, shown above, is a game where you only read and type what has been set.

**Washing Machine**

Another example we can use for EDP language to be suitable is washing machine. The input for a washing machine is to put the clothes into the slot. Once this is complete, the washing stops with the spinning. Another human interaction is when the user chooses what to do after it. For example, spinning the clothes.

* Setting a timer
* Pressing a button
* Rinse clothing
* Drying it

These are some of the features have been included. Event-driven programming are suitable to the features that have been named. The main events that take place within EDP is button clicking, mouse movement, key pressers, timers, and sensors. All of these have included within the washing machine.

**Conclusion**

In conclusion, I feel that only some parts are suitable for event-driven programming to be implemented. The main reason is that it is suitable for graphical application, because the user can see the graphical interface. With non-graphical, it is similar; but using a washing machine for EDP would work but not properly. I feel that non-graphical applications with graphical applications are suitable. It suits more towards graphical applications.

**D1 - Reference**

<http://ubuntuforums.org/showthread.php?t=1937327>

<https://prezi.com/boitgjmuue42/key-features-of-event-driven-programming/>

<http://liton.unit20.assignment1.angelfire.com/d1.html>