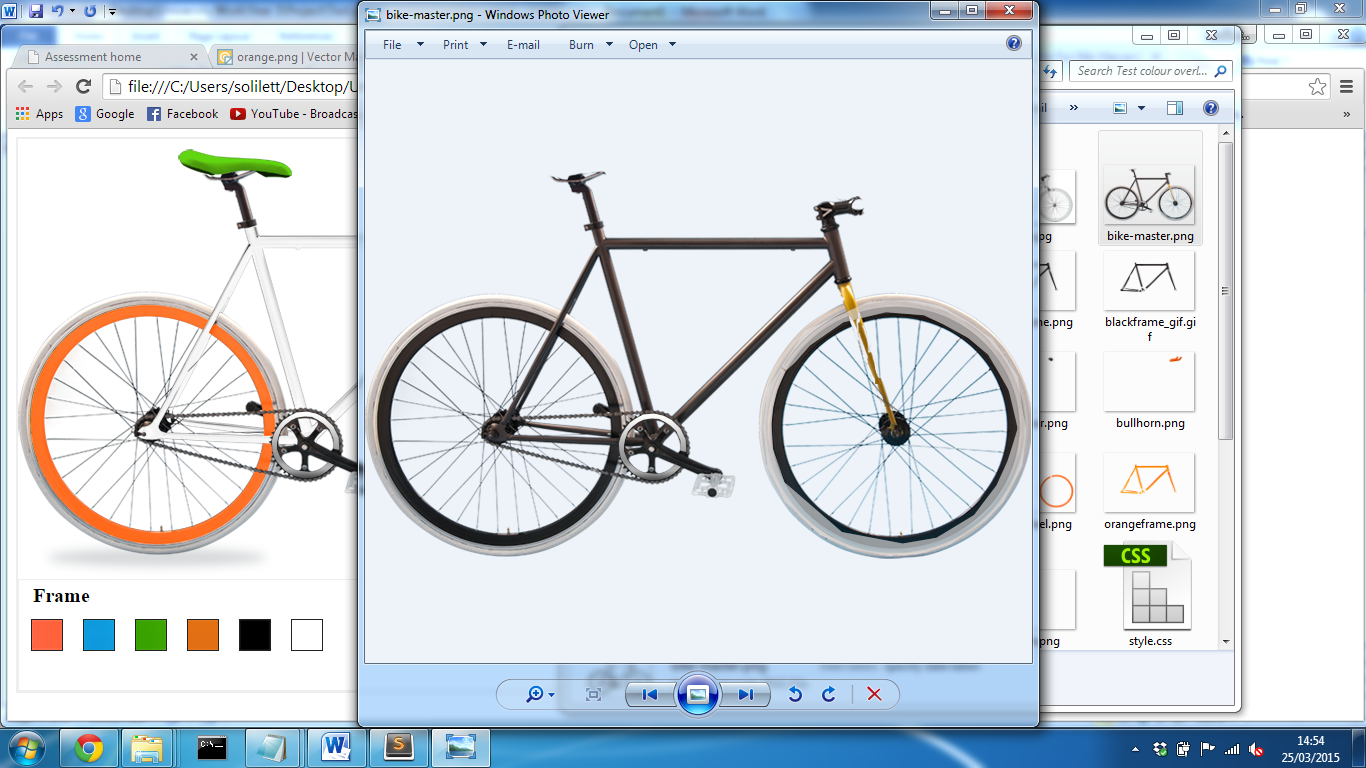
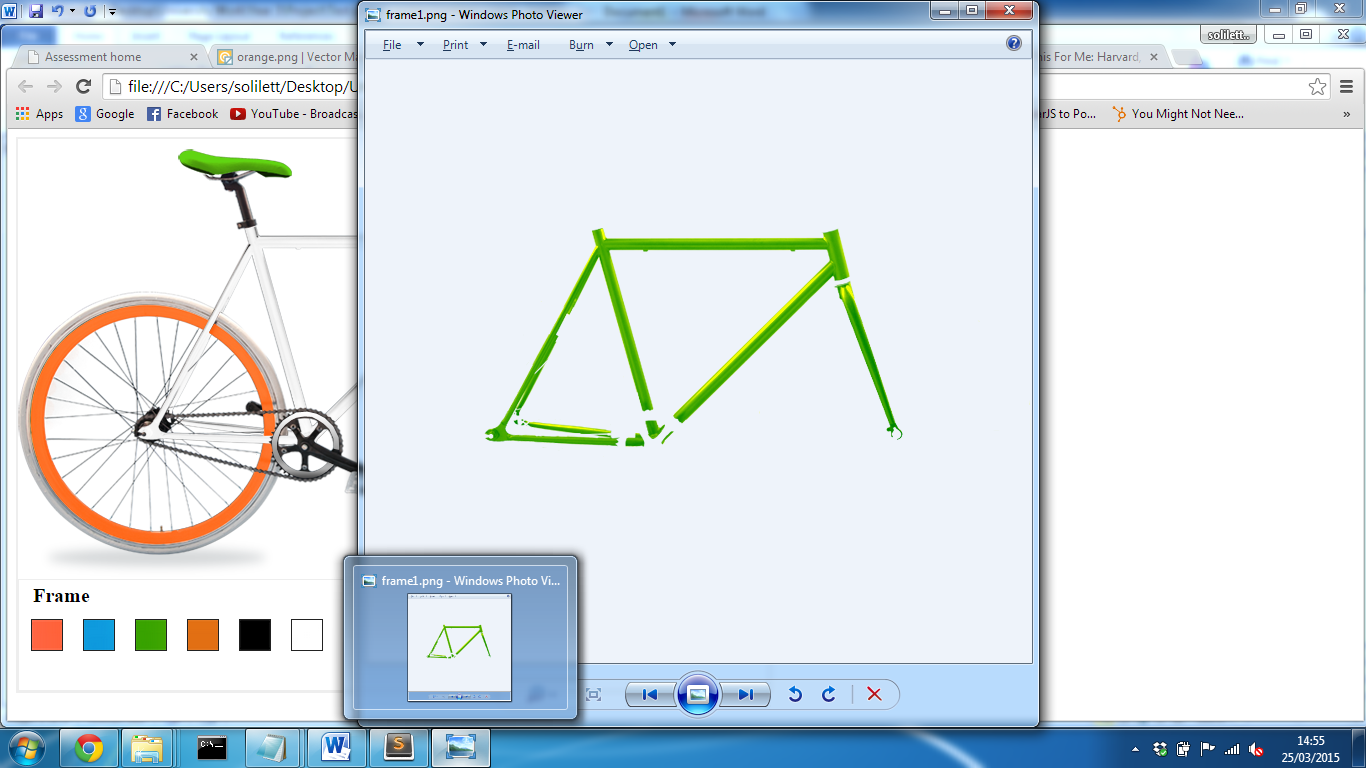
**Proof of Concept**

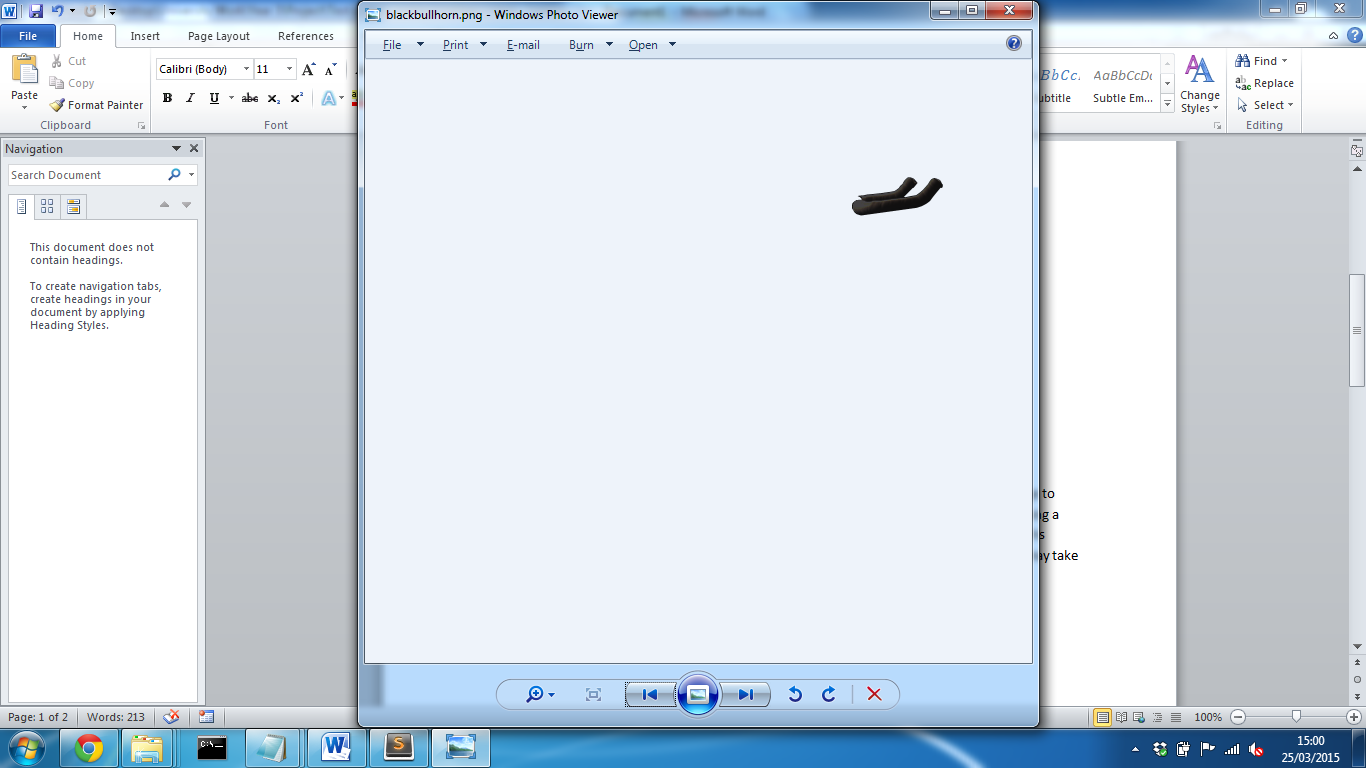
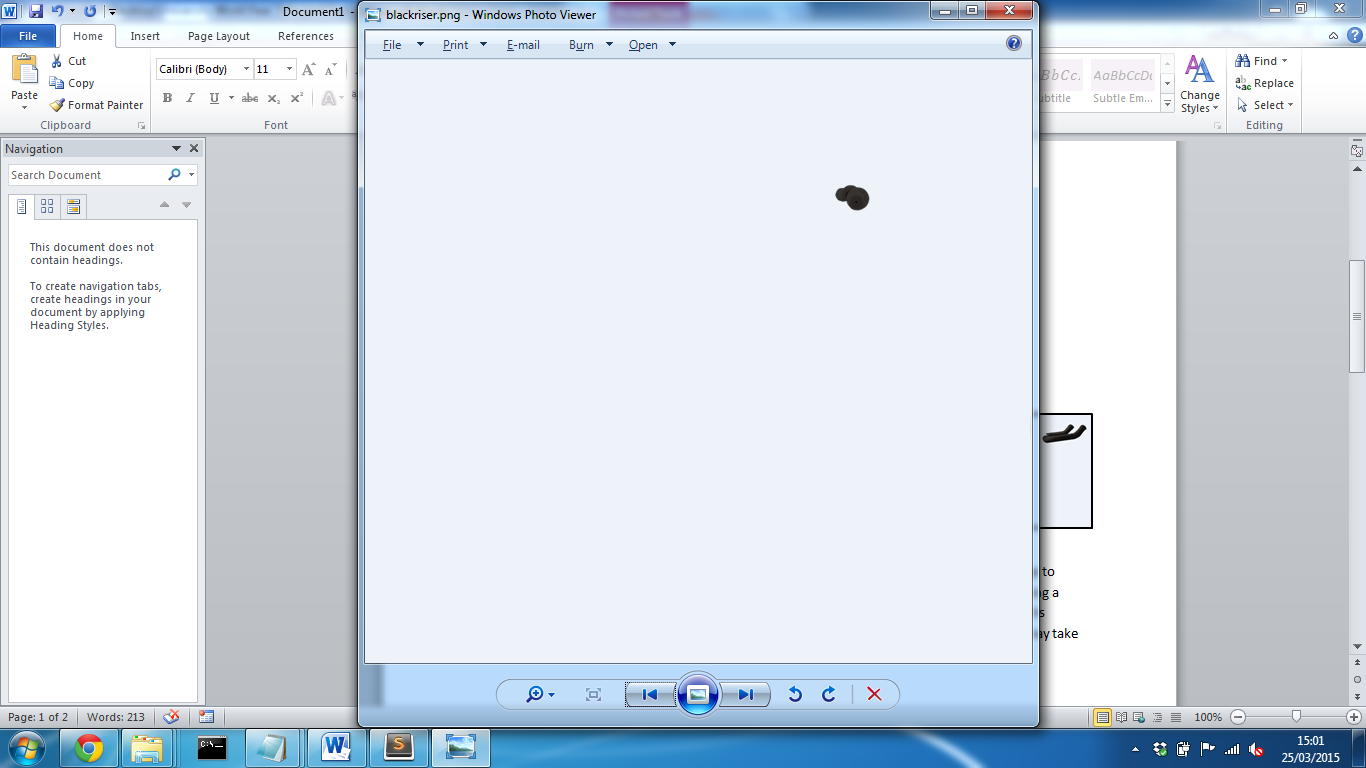
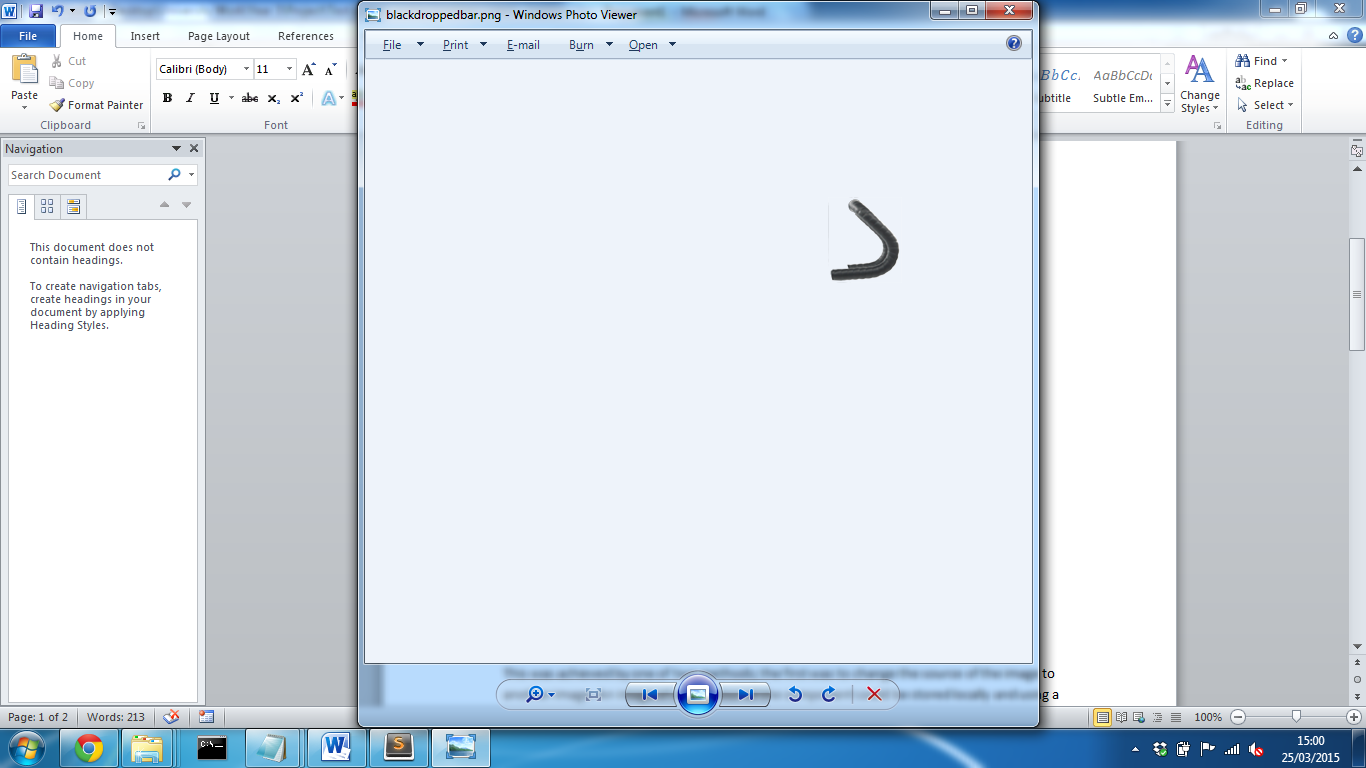
Before development of the final artefact began, the initial stage followed the proof of concept method, which demonstrates the feasibility of the desired system. This method allows a prototype to be created to ensure that the final artefact requirements are reach, but doesn’t provide a deliverable. (Janssen, 2015)  
This meant the first development task was to implement a working solution that allowed an image to be manipulated, providing the illusion the cycle component was changing colour.   
In order to allow each component to be modified, the cycle needed to be split up into each component that the user can interact with. The cycle images consisted of the following images:

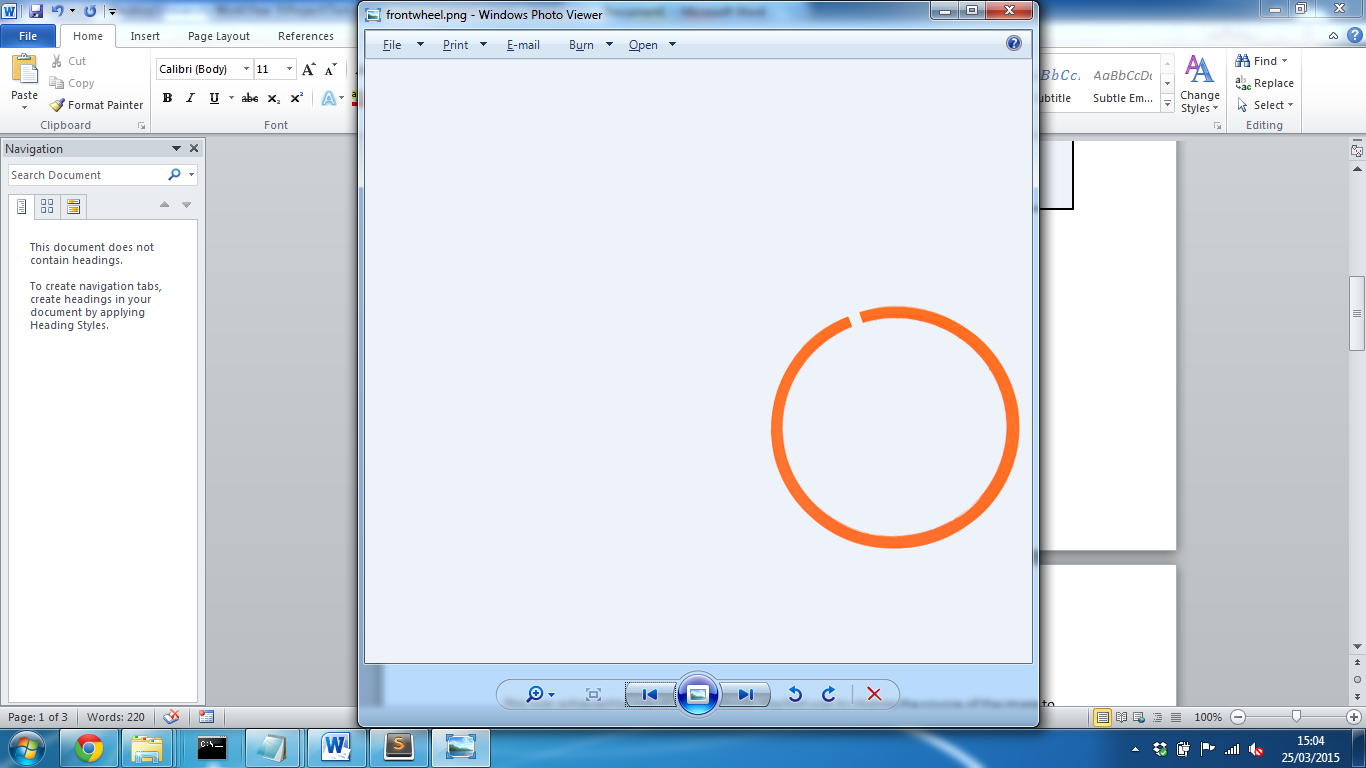
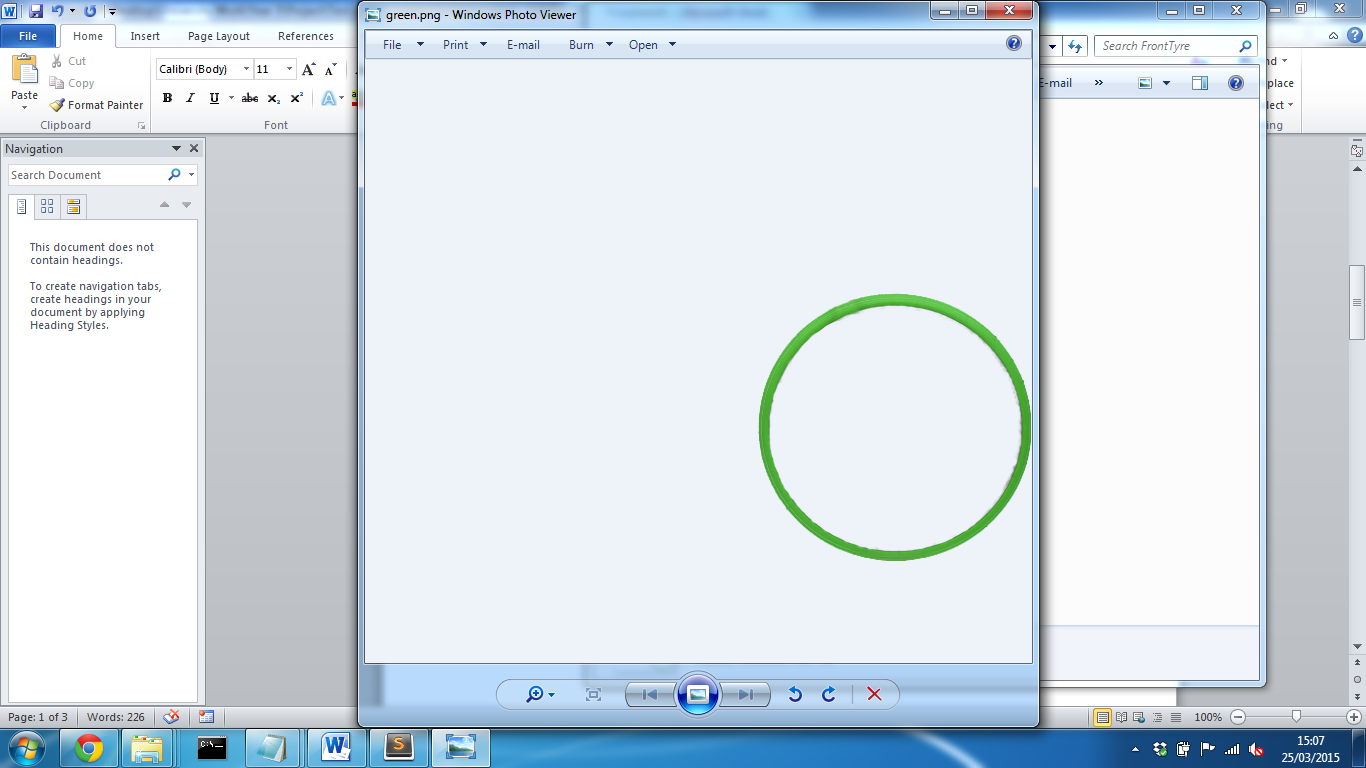
* Master Image



* Frame



* Handlebars - Each type: riser, dropped and bullhorn
* Wheels and Type (Front Tyre)



Each component was edited using Adobe Photoshop and then cut from the entire master image. It was the placed onto a blank canvas with the same dimensions as the master image in the same position. Photoshop allows for part of the image to be traced based on its colour, in perspective to the surrounding colours i.e. Background. After selecting the image can then be placed into a new canvas and Photoshop will automatically place the selected image in the same location as the other canvas, if the new canvas is of the same dimensions. Each image could then be masked and a new colour could then be overlayed on top; this meant that the colour could be changed and the shades of the original image could still be present to create a realistic image. Each image was saved as a PNG, which allows the background to be transparent.

After each component had been extracted from the master and saved in a structured folder layout, the cycle could then be implemented into a webpage. In order to create the illusion that all components are one image of a cycle, they needed to be layered on top of one another.

<img id="master" class=”component” src="bike-master.png" />

<img id="frame" class=”component” src="frame/green.png"/>

<img id="frontwheel" class=”component” src=”frontWheel/orange.png" />

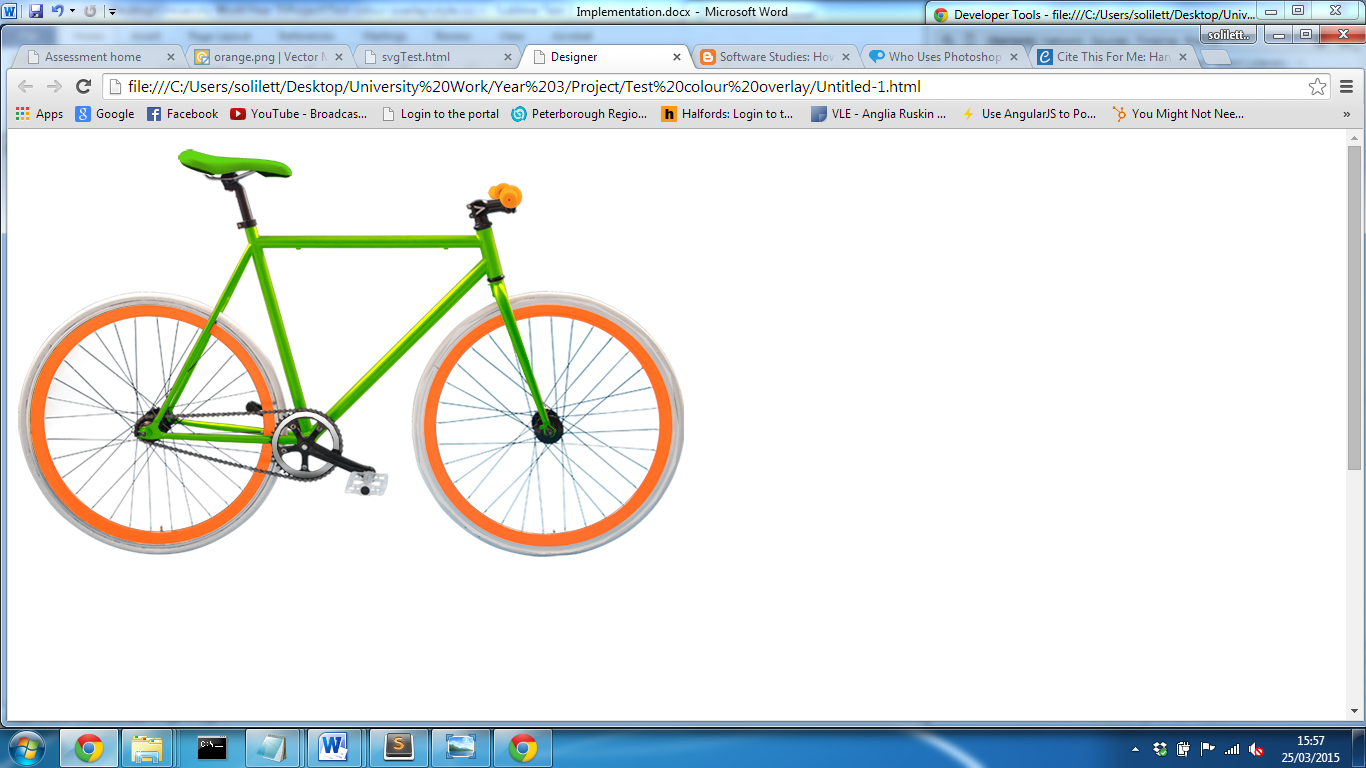
<img id="frontTyre" class=”component” src=”frontTyre/white.png" />

<img id="rearwheel" class=”component” src="rearwheel/orange.png" />

<img id="rearTyre" class=”component” src="rearTyre/white.png" />

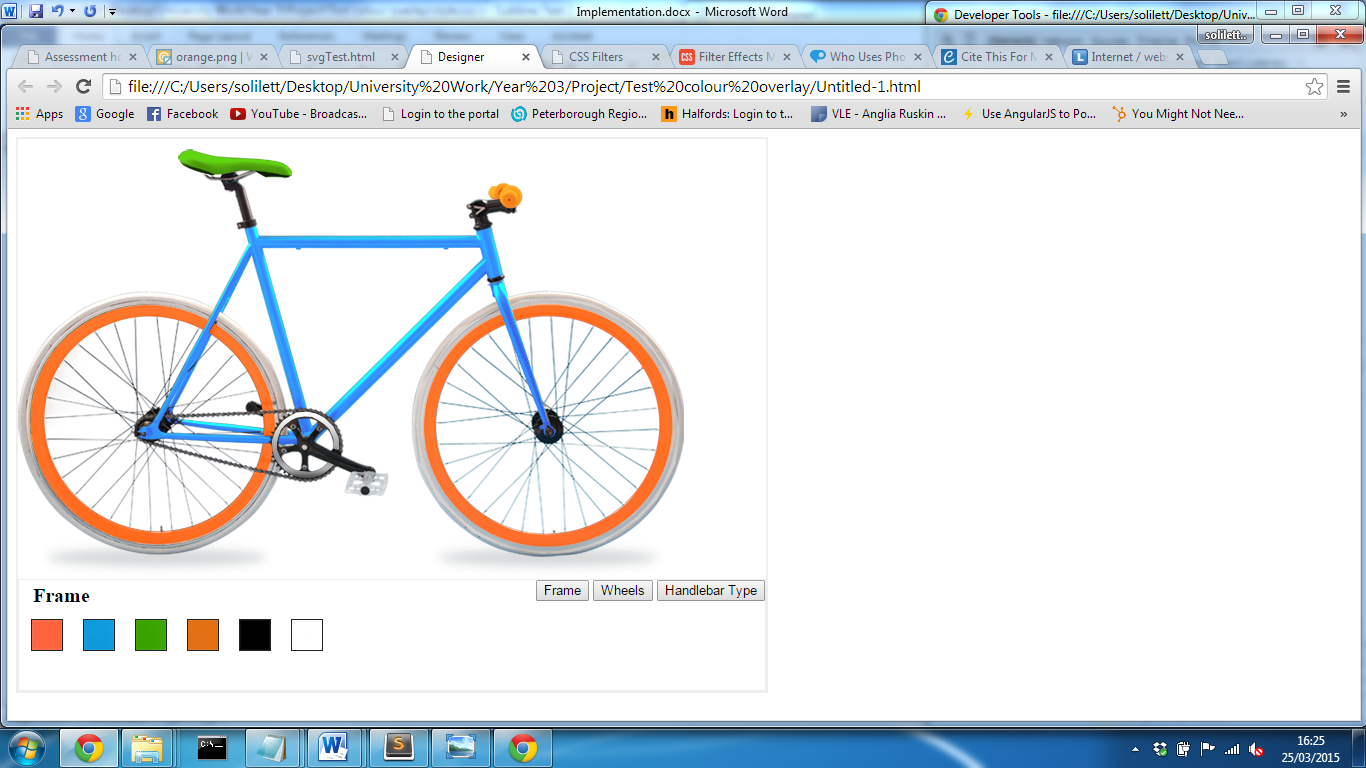
<img id="handlebar" class=”component” src="riser/orange.png" />

<img id="saddle" class=”component” src="saddle/green.png" />



The class “component” was then set to “position:absolute” which positions all components in the same place. The z-index of each component could then be set so that the master was at the back and components where then layered on top.

The next task to be implemented was the user interaction to allow a component colour to be changed. This was achieved by one of two methods; the first was to change the source of the image to another image. Using a JavaScript library the image source could be modified which would display a different colour component. While this was a feasible option, this meant that a large number images would need to be stored; it also meant that content may take some time to load.   
The second option was to use filter effects; these are a way of processing an element’s rendering before it’s displayed to the user. (Dev.w3.org, 2015) The Hue rotation filter allows for a degrees parameter to be specified between 0 and 360; any value between these values will then render a different colour to the user. The hue rotation value can be applied to a specified element through the css.

**CSS**

#frame {

webkit-filter:hue-rotate(120deg);

}

By setting the hue rotate parameter to 120deg allows the frame to transform from green to blue. While this works extremely well for a multitude of colours, black and white colours cannot be achieved. Although other filters are available, one filter alone will not achieve these colours and only one filter can be applied to an element. In order to overcome this problem, both methods can be used; this will mean that when black or white is selected the image source change and when any other colour is the hue rotation filter will be applied. While this reduces the amount files stored, user experience could be affected as the user may notice the time taken for the image to render. This method would be used in the final solution as when changing the handle bar type, there would be no other option.

After accomplishing a method to manipulate the colour, a function needed to be implemented that allowed for user interaction. In order to achieve this, a set of colour tiles were displayed to user and jQuery library was used to handle events. Using the jQuery library, a large array of pre-determined JavaScript functions are accessible; while JavaScript alone would be suitable, based on the developer’s previous experience, jQuery required far less lines of code.

$("#frameTileBlue").click(function(){

$("#frame").attr("src","frame/orange.png");

$("#frame").css("-webkit-filter","hue-rotate(120deg)");

});  
…

Using the jQuery library, a function can be specified based on a certain event. The above code will, on the click of an image with the id of “frameTileBlue”, will run a function to set the source of the images with the id “frame” to orange.png and apply the hue rotate filter to turn the frame blue.

To see implementation of the proof of concept prototype find video “proof of concept.avi” in videos folder.

References

Janssen, C. (2015). *What is a Proof of Concept (POC)? - Definition from Techopedia*. [online] Techopedia.com. Available at: http://www.techopedia.com/definition/4066/proof-of-concept-poc [Accessed 25 Mar. 2015].

Dev.w3.org, (2015). *Filter Effects Module Level 1*. [online] Available at: http://dev.w3.org/fxtf/filters/ [Accessed 25 Mar. 2015].