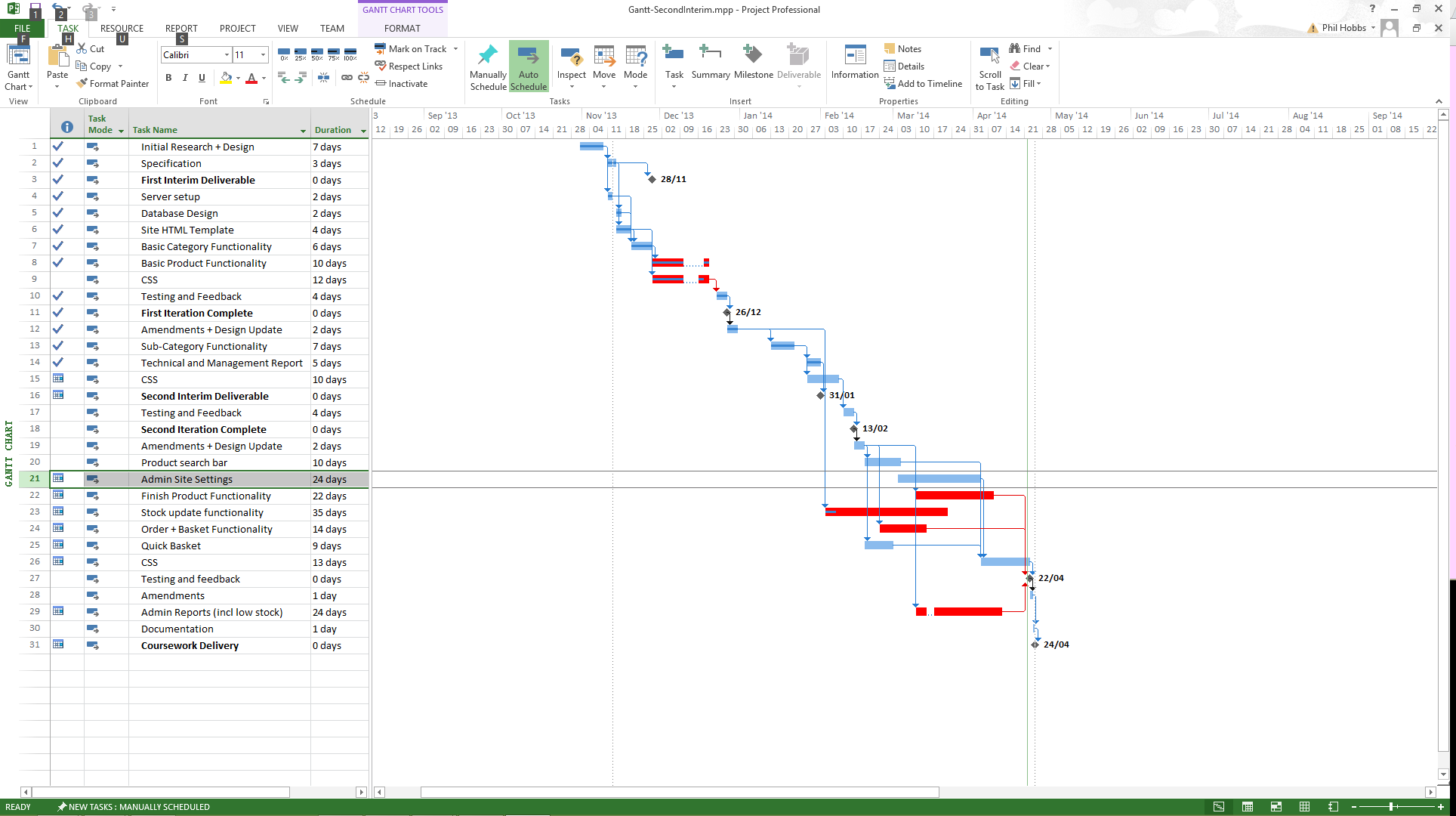
Project Report

# Lifecycle and Schedule

The project started well but gradually began to drift further and further out of line with the proposed schedule. At the time of the second interim the project was only slightly behind schedule and I had made what I felt to be appropriate changes to the schedule to include all the desired features in the final product. However in reality the appropriate action would have been to prioritise more the essential parts of the specification and drop out some of the more superfluous features.

Initially following the principles of DSDM (Dynamic Systems Development Method) I had planned to fully implement the basic features and add incrementally the less necessary but more user experience enhancing features. But I was distracted adding ‘extra’ features and failed to efficiently include all the basic features I planned to.

As a result my attempt at planning became somewhat of a mess and horribly inefficient, as my Gantt chart below should reflect:



My initial project plan and Gantt chart have become somewhat irrelevant in this phase as the majority of my time since the previous interim has been spent on the small details rather than creating a fully functional product and then the final few weeks were a mad rush to try and get some basic features added in. Following the initial plan to complete would have help prioritise better.

The Gantt chart shows various tasks moved from the second iteration to the third. Upon reflection, too many things were added to the project that were not sufficiently foreseen, and also many basic features were overlooked and omitted from the original schedule and even updated schedule from the second interim.

Core functionality such as display full product information on individual pages, much of the CMS management and all of the admin were left until the last iteration to implement. Unsurprisingly this caused major complications.

# Surprises and Complications

There were many of these along the way, some of which have already been picked up on interim reports. However many of the worst of these were in the later stages of the project.

* Getting lost in code!
  + Simple tasks took much longer when fighting my way through lines of code.
  + Learning process resulted in probably horribly inefficient code, inconsistent use of code, and having to rewrite code multiple times.
  + Although not usually one for omitting comments, I had initially slacked somewhat on this which resulted in the daunting prospect of going back through lines of code adding comments when I was already behind on the delivering an actual fully functional product. Needless to say, it ultimately didn’t happen for much of the code.
* Implementing basic functionality of custom product attributes was much simpler than expected, as was implementing an AJAX image uploader.
  + Although I did not implement very advanced versions of these features, I was relatively happy with the result of both.
* Implementing ordering with search features and pagination was quite difficult.
  + I had initially planned to allow products to be ordered on the main shopping page, however I struggled with how to implement this with searching enabled and how to use the results of the search in an order.
  + I ran out of time to implement pagination because I had wasted too much time on features such as touch control (which I was unable to successfully implement) and small details such as using SVGs for almost all the icons on the site, and so had no time to research it.

# Lessons Learnt

* Prioritise basic functionality over aesthetic and cosmetic features.
  + Too much time wasted on drawing up SVG icons and learning the technology as well as attempting to implement drag and touch features and responsive design.
  + Ran out of time to implement more basic and certainly more fundamental features such as improved admin order management and pagination of shop products.
* Modularise\Document\Design code.
  + I didn’t realise how much code I would end up writing due to my limited experience with web projects and it quickly became unmanageable and took me twice as long to accomplish anything. Better modularisation of the code and files would have help me to make more effective use and reuse of the code.
  + Documentation speaks for itself, literally. Trying to remember what everything does without it is very hard once you start writing a lot of code.
  + By “design,” I mean actually plan out and design how code is going to look beforehand. I believe this will help to avoid inconsistencies in code and confusion when dealing with existing code particularly.
* Don’t be overly ambitious, be realistic
  + My ambitions for this project were great; my expectations were somewhere in the middle and my product was left behind in excitement. As I started learning many great new things I wanted to put everything in my project and even wasted days researching specific things that never made it into the final code.
  + If I had had the same length of time with far fewer other commitments I would have been able to accomplish something closer to what I wanted. Unfortunately this was not the case, although I thought I had accounted for it, I was a long way off doing sufficiently.

## References

The following were used in my project:

Font Awesome - <http://fortawesome.github.io/Font-Awesome/>

Droid Sans - <https://www.google.com/fonts/specimen/Droid+Sans>

Due to experiencing some issues with fonts and icons not displaying as a result of a server error I downloaded the source files for both Font Awesome and Droid Sans font to use locally rather than via the provided CDNs.