Specification and goals:

The project specification was to create a Rubik's Cube solver. It was expected that initially the Simplest Solver, or Beginner's Method, should be implemented and then other methods should be compared to this. The specification suggested finding an optimum or lowest moves solution in a reasonable time frame by possibly considering parallelisation. This could then be extended by looking at different sized cubes or by displaying the solution to the user. I attempted a quick solution instead of an optimal one. Initially, the Beginner's Method was used as this had been the quickest method in the past and provided a good comparison. This was expanded by comparing and implementing two more methods and parallelising them. These methods were iteratively improved throughout the project in an attempt to improve the solve time. The goal of the project became creating the quickest software solver for a Rubik's Cube.

Previously the quickest average solve time for the Rubik's cube in a software implementation was 39.1ms. I achieved this with the first implementation of a method. The first implementation was using the Beginner's Method but was not as efficient as it could have been, it took 1.54ms to solve on average with this method. With the initial goal of finding the quickest solve met the rest of the project continued to try and improve on this solve time. I did this first by trying different methods that seemed like they could have lower solve times and then improving upon them.

With improvements, Roux was taking 0.551ms to solve on average and the Beginner's Method was taking 0.553ms. Finally, I parallelised the methods. This effectively took the best solve of six. As a result, there were reductions in solve time with the best taking 0.423ms to solve. This means the goal of the project has been achieved and improved on substantially. Further parallelisation may be possible but would require some sections of the program to be rewritten. If I had considered this from the start I could have tailored my program to this but did not as I only properly considered parallelisation later on.

To achieve these times there were sacrifices. The moves to solve were substantially higher than expected for the implemented methods and were much higher than the optimum number of moves to solve. The Beginner's method took 187 moves to solve on average after parallelisation which is much higher than the expected value for the method of around 110. The other methods saw larger differences. It is possible this was required to get a quick solve time and that low moves may not be achievable with a low solve time. I think this was a necessary sacrifice to achieve the quickest solve.

Self-reflection and Skill Development:

Problem Solving: In the literature review of the project, I determined which Rubik's Cube solving methods would provide a low solve time. This seemed like a difficult task. With hundreds of solving methods, it was hard to determine how quick they would be by going through the methods. As a result, I tried to break the task down. I came up with a way to evaluate methods that could be worth trying. This may not have been the best way but showed me the power of breaking down a task to make a problem much easier to manage. This will make it easier for me to tackle large problems in the future.

The software would crash in the beginning. This presented quite a difficult bug to find in the code which required problem solving. The program was only crashing on 1 in 1000 solves so the issue was difficult to isolate. When the code was smaller issues could be found by looking through. As it grew this was no longer practical. I learned to use the debugging feature of Visual Studio. Allowing me to see the line of code or loop the program was stuck on making problems much easier to identify. Sometimes this still provided a big area to check with large loops. Using current variable values I was able to see how the values changed as I stepped through the code. Allowing me to see what was causing problems. I have improved at debugging which will be key to solving problems in my code in the future.

Communication: Throughout the project, I have received feedback at various stages of the project. My supervisor identified areas of my report where my written communication was not as clear as it could have been. I have tried to take this feedback into account to make my communication clearer and to write better in the future. This will be especially useful writing formally as my supervisor has informed me of conventions that I was previously unaware of.

Responsibility: In the past, I have never taken on such a large project. I have had to adapt the way I normally manage my time when working. I spread the project out allowing me to fit both project and other university work into a week and to work on the project at a much more sustainable pace. I had a plan for work for each day of the week. This has allowed me to slowly adapt and develop my project and has given me time to improve the project further than its initial goal. If I had worked on the project too much at once I may have burnt out and if I had started too late or had been less organised I may not have been able to manage these improvements. I have learnt the benefits of good organisation and tackling a project sustainably. In the future, it will be useful to take these things into account and allow possible improvements in different work.