While playing darts with my brother during the first lockdown, I realized that the laborious task of keeping the score could be made significantly easier by writing a simple program that could do the job for us. Following YouTube and Medium tutorials, I learned the basics of frontend development using HTML, CSS and Javascript and created a working web page. This motivated me to apply for work experience at ###, where I was introduced to backend development for the first time. Using React.js and Node.js together with MySQL, I built more complex webpages, such as a large-scale e-commerce site for my "Young Enterprise" company. The reason why I chose React.js was because I found the hierarchical structure of the codes and files very intuitive and useful compared to other frameworks. Another framework that I used was Django, where I got introduced to Python

As I quite enjoyed Django, I attended a course given by AI Core on the basics of Python in August 2020, where I learned about the syntax as well as simple machine learning algorithms like categorising data using various regression models. Utilizing what I learned as well as following online tutorials, I built my first image classification NN (Neural Network) using the MNIST dataset. By logging the data using Tensorboard for different layer sizes, activation functions, and locations of Dropouts etc, and by reading articles such as "Simple Introduction to Convolutional Neural Networks" by Matthew Stewart, I gained a solid understanding of the theory behind NNs. Along with image classifications of cats vs dogs and car vs bike, I also attempted to create a model classifying street signs. While researching for this project, I read quite a bit about RL (Reinforcement Learning). As I reckoned that I had to work on smaller and simpler projects to learn the basics of RL before I attempted my dream, an autonomous car model, I used many different OpenAI's environments to understand how to use the given observations and creating a functioning neural network. During this process I experimented with lots of different policies and algorithms, figuring out their strengths and weaknesses. When I was confident that I understood how to build RL models, I switched over to Unity and its new ML-Agents package as I had complete freedom on creating my own environment. The most significant hurdle I had to overcome was working with C#, a programming language that I never attempted before. Thanks to YouTube tutorials and great resources from Unity, I familiarized myself with the language unexpectedly quickly, but encountered numerous problems while creating the autonomous car model. OpenAI always simplified actions, observations and rewards while in Unity, I had to create and process my own observations, needed to decide whether the observations and actions should be discreet or continuous, and where to give what reward. Many attempts ended up with very slow, or no progress at all. After finding a good balance, I sped up the process by starting off the training with imitation learning with my own inputs. That way I successfully managed to create an autonomous car model.

Apart from NNs, I also really enjoy modelling maths and physics. Due to this, I spent a long time building a realistic simulation of rockets, including varying gravity, and resistance forces at different altitudes and latitudes. The grc.nasa.gov website was a great steppingstone for deriving equations such as thrust and air resistance. The most challenging part of this project was finding a way to animate Matplotlib canvases using call-backs. I found that the best method was using the FuncAnimation call-back class together with a Line2D object as the rocket. This experience also helped me tremendously during my school's "Lockdown Coding Challenge", which tasked us with creating the most accurate 3D model of the Solar System. Together with a friend, we successfully completed it after being stuck on multiple occasions. One of those was working out the best way to calculate the mean anomaly and hence the sweep of the bodies around the sun using Kepler's 2nd law. Utilizing the skills that I learned in Further Mathematics such as implicit differentiation, we ended up with a very accurate model. A few weeks later, I stumbled across a YouTube video on turbulence by '3Blue1Brown', which inspired me to create a 2D and 3D model showing turbulence of incompressible fluids in Objective C using the Processing IDE. This stretched my physics, programming skills, and researching skills beyond my limit,

but it ended up great and I learned a lot, especially by broadening my scope of C languages which is useful for future projects. Since then, I started working on visualizations of laminar and turbulent flows interacting with solid objects.

Apart from working on big projects, smaller challenges with a specific target, such as reducing time complexity or using unfamiliar modules, helped me understand and improve gaps in my understanding. LRU caching and recurrent backtracking have now become my favourite algorithms despite being completely unfamiliar with them before. The most valuable piece of code I wrote in the process however was the quicksort algorithm without prior knowledge of it, with a worst-case time complexity of O(n logn). This algorithm has helped me in many other projects, especially for large datasets for analytics. At an internship at SAP, one of my tasks was creating a presentation of these algorithms for Africa Code Week, which was not only very enjoyable, but also helped me understand them better. Apart from that, I also worked on fixing bugs and errors within the SAP BTP trial system, which was exiting as this was the first time interacting with cloud computing, and it taught me a lot about the future of IT.

At university, my goal is to improve at a wide range of topics, so that I can one day fulfil my dream of building an autonomous system for a real car.