# Final Year Project Demonstration

Wardrobe Buddy -

A software engineering project.

# Introduction and Motivations

I have developed a full stack application with a fully developed front and back end in Swift for an iOS application. The app I have developed allows users to visualize their clothes in different ways and suggests outfits using inspiration from the Gale-Shapely algorithm. The application uses an element of computer vision to recognise an object and its primary colours, while also using an object detection model to detect and recognise either tops bottoms or shoes. It then uses a similarity matrix to closely match the images to each other and then to the style in a 2-step process.

The motivation of this project came from my indecision to choosing what to wear every day. I found myself wearing the same things all the time while having a vast wardrobe and knew I wanted to change this. I did some research and according to a study conducted by Marks and Spencer's that for the average person in the UK they only wear 44% of the items in their wardrobe. This showed me that there is scope for an app to solve this universal problem.

There are several existing solutions out there such as Uniqlo's outfit generator where you can build an outfit from the new pieces in the collection. Or there are apps such as Essembl where you can manually upload outfits. My implementation is different as it takes inspiration from both apps however it allows the user to create outfits on an outfit-by-outfit basis based on what is available in their wardrobe.

# Key aspects of the project

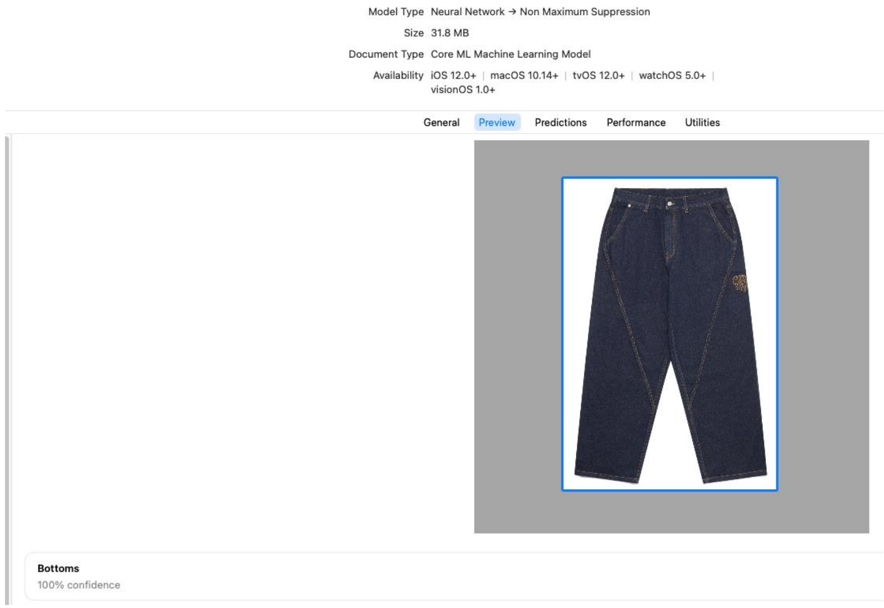
## Firebase

The backend for this project has been set up using the Firebase API. This is a google backed database and it is very simple to implement into our project as we just add the packages via copy bundle resource in build phases in XCode. We need to add the GoogleService-Info.plist. For my application I have used authentication to enable user authentication and also user stability. Leveraging the authentication aspect we then create a document using the users unique user ID. We create 2 fields, photoArray and outfits. PhotoArray stores variables such as the brand, type, size, rgbValues, rejectionCounter, url and timestamp. Where outfits stores each saved or favourited outfit as its own document which consits of top bottom and shoe. When the user uploads an image it is saved to the Storage part which keeps all of the images in separate files for each user. This is called upon throughout the project via live listeners.

# Key aspects of the project

## Object detection

I used a neural network which utilises non-maximum supression for machine learning for object detection. This model needed to be trained on a small-medium dataset. There are 3 classes and for each class it was trained on 85 images per class which totals to 255 images in total. Out of 25 test images the model correctly predicted 100% of the images with an average confidence of 96%.



Model Type Neural Network → Non Maximum Suppression

Size 31.8 MB

Document Type Core ML Machine Learning Model

Availability iOS 12.0+ | macOS 10.14+ | tvOS 12.0+ | watchOS 5.0+ | visionOS 1.0+

**Bottoms**

100% confidence

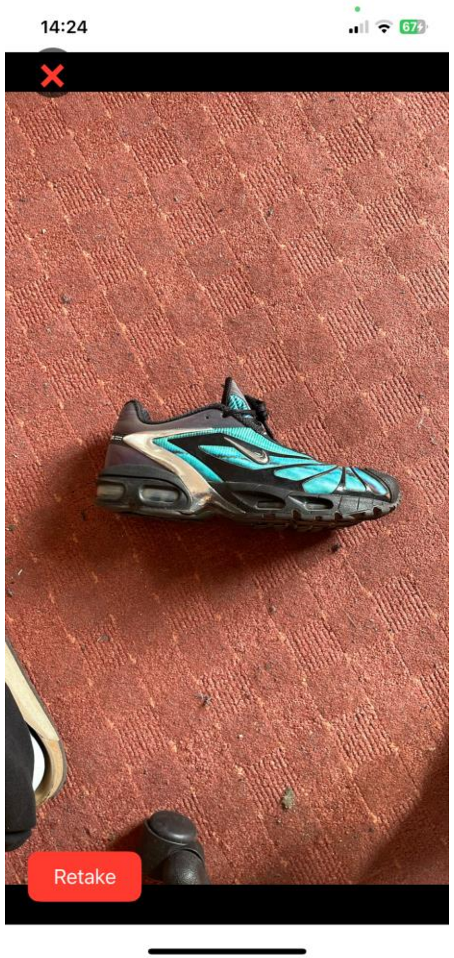
## Colour Detection

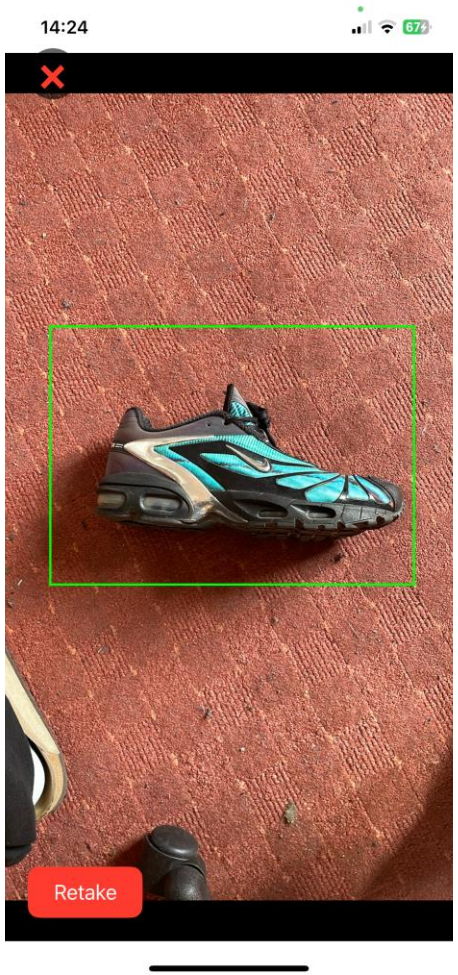
Another key aspect is colour detection when the user uploads images we calculate the predominant RGB values within the image once all of the colors have been mixed together.

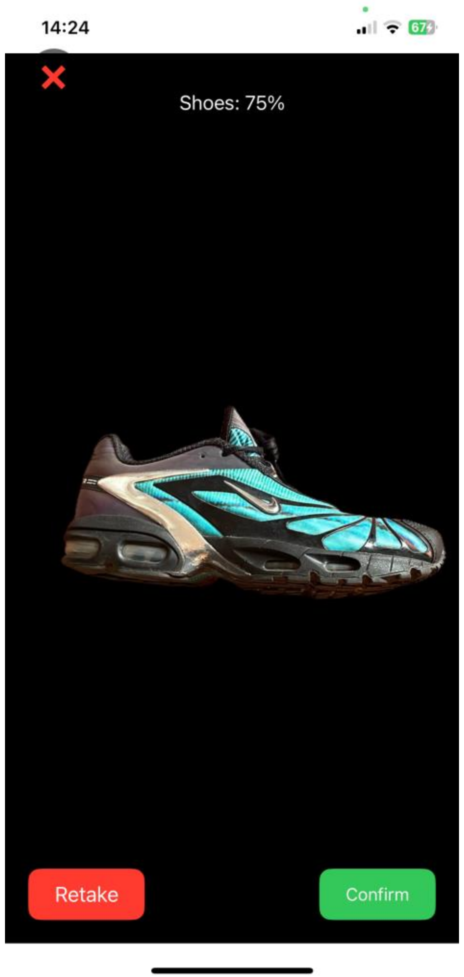
# Key aspects of the project

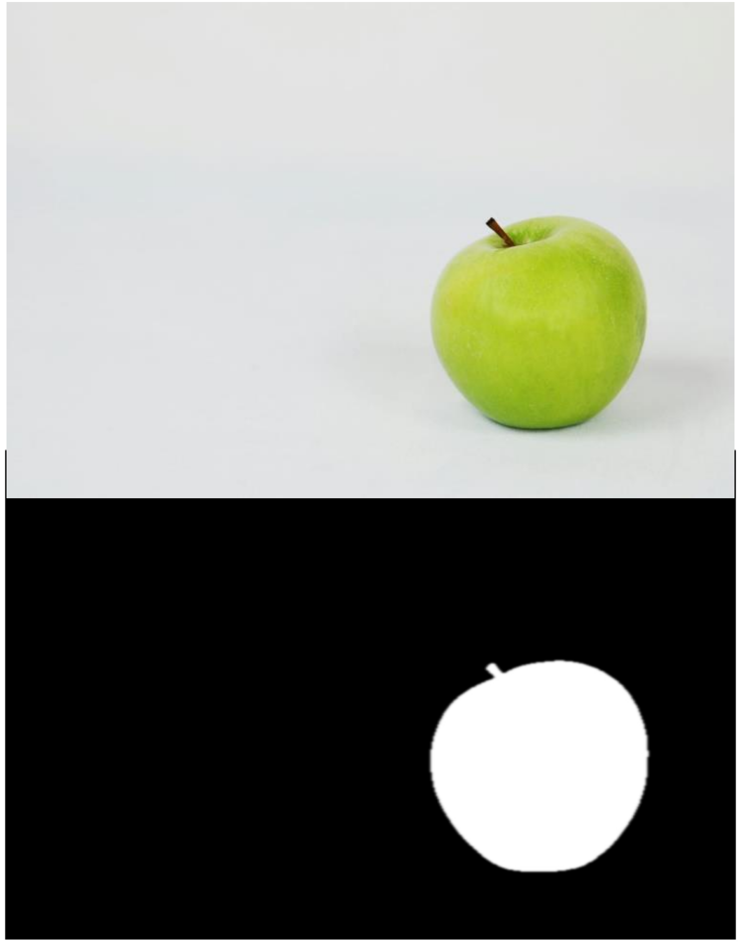
## Background removal

This was a fundamental feature of the uploading image process as the background for an outfit is irrelevant. We use a class within the Apple vision framework which is a request that generates a mask of noticeable objects to then remove the background. So we take the photo that the user has just captured and we remove the background from the bounding box that the user draws.









# Key aspects of the project

## Outfit recommendation.

For the outfit recommendation I took inspiration from the Gale-Shapely algorithm where it is a propose and reject algorithm however I have not directly implemented it. Firstly the user chooses a style which has been preset to various colour properties such as brighter, darker more tonal etc. Firslty we use a similarity matrix to compare how similar the items are with each other. We calculate the similarity and the contrast here. Then each potential combination recieves a score based on how the combination matches the selected style. The score determines in which order that combination will be shown. This has been replicated as an optimisation problem where we want to see the best outfits at all times.