Final Year Project Report

Augmented Reality Application

“WeAR It”

Oliver Nagy - 19269749

Table of Contents

1. Introduction

1.1 Overview

My augmented reality app allows users to virtually try on different pieces of clothing, such as shirts, shoes, and trousers, using their smartphone camera. A Designer can upload their designs for users to try on, giving them a better idea of what the clothing will look like on them before they make a purchase. The aim of the app is to provide users with a less stressful and more enjoyable shopping experience, while also giving designers and businesses the opportunity to market and sell clothing in a more innovative way. Users can easily browse through a variety of clothing options and see how they look on them in real time, all from the comfort of their own home. Whether you're a designer looking to showcase your creations or a shopper looking for a convenient and enjoyable way to shop for clothing.

This app would try to eliminate the two major reasons why online bought clothes get returned which are "Items don't fit well" & Items don't suit me" [1]

This app would help reduce returns from customers /users which have a negative impact on the environment. [2]

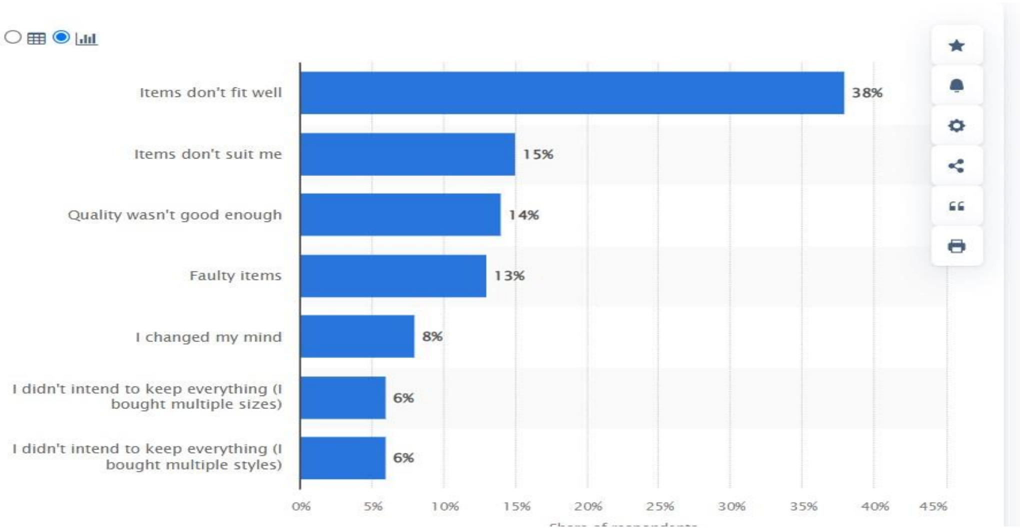


Fig 1. Chart depicting why clothes get returned [2]

1.2 Introduction to Augmented Reality

Augmented reality (AR) is a technology that overlays digital information and virtual objects onto the real world, enhancing and augmenting our perception of reality. AR can be experienced through various devices such as smartphones, tablets, smart glasses, and head-mounted displays. It works by using sensors, cameras, and software to track the position and orientation of the device, and then superimposing digital content onto the user's view of the real world. There are two main types of AR: marker-based and markerless. Marker-based AR uses a specific image or object as a trigger to display digital content, while markerless AR uses real-time mapping and tracking of the user's environment to anchor virtual content in the real world. AR has a wide range of applications in industries such as education, entertainment, gaming, retail, healthcare, and more. It can be used for anything from virtual try-ons in fashion to medical simulations and training. Overall, AR offers an immersive and interactive experience that blurs the line between the digital and physical worlds. [3]

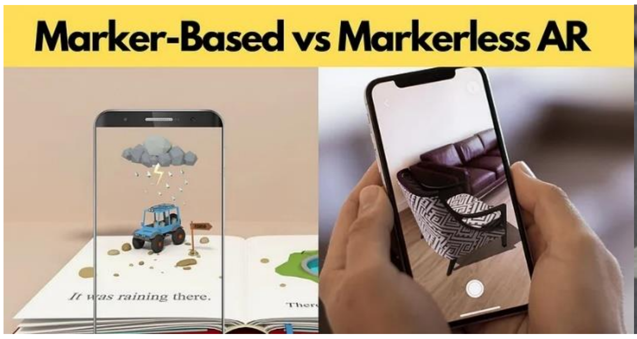


Fig.2 Image showing the difference between marker based and Markerless AR [4]

Fig.3 Image showcasing an example of an AR application [5]

2. Market Assessment

During my market research, I have found several similar applications that focus specifically on augmented reality (AR) for clothing. These include Adidas, SneakerKit, WannaKicks, and a Zara AR app concept developed by Supersuper agency.

While these applications all allow users to view and interact with shoes in AR, there is also a Burberry AR collaboration with Google that does not allow users to try on clothes, but rather just view them in AR, meaning you can place objects in a room but not try them on.

2.1 YourFit

Another app I found during my research was YourFit. YourFit by 3DLook is an augmented reality (AR) app that allows users to virtually try on any clothing using their smartphone camera. According to the company's website, the app uses advanced 3D scanning technology to accurately capture the user's body measurements and fit preferences. Here's how it works:

The customer uses the app to take two photos of themselves, which are then processed using 3D scanning technology to create a 3D avatar of the customer. The customer selects a garment they would like to try on from a list of available options in the app. The app uses machine learning algorithms to adjust the garment to fit the customer's 3D avatar in real-time. The customer can view the garment on their 3D avatar in an augmented reality setting, allowing them to see how it looks and fits in real-time. However, upon downloading and trying the app, it appears that only a limited selection of clothing is available, with a focus on options for women. This may indicate that the app is still in the early stages of development and may expand its offerings in the future. You also need to request a demo on their website to use it therefore it's not publicly available yet.[6]

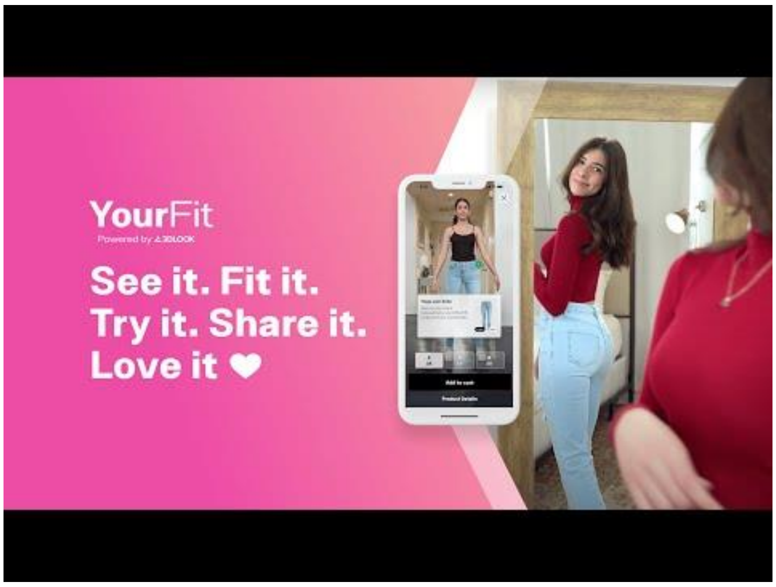


Fig.4 Video demonstration of YourFit [6]

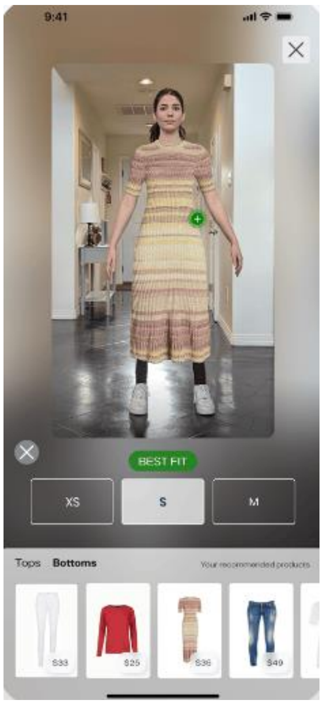


Fig.5 Screenshot from the YourFit app [7]

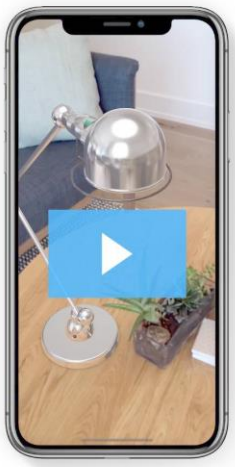
2.2 Shopify

Shopify have also introduced an augmented reality feature called AR Try-on , although the app doesn't allow you to try on clothes it does let the user upload their own images of the items they are selling after which the person looking to buy it can experience it in AR. The app uses the ARKit framework for iOS devices and ARCore for Android devices. customers to view 3D models of products in their real-world environment. Customers can use the app to visualize how the product would look in their space before making a purchase.

The AR Try-On app by Shopify uses ARKit and ARCore to provide an immersive and seamless augmented reality experience for customers. ARKit is a platform developed by Apple that enables developers to create augmented reality experiences for iOS devices, while ARCore is a platform developed by Google that enables developers to create augmented reality experiences for Android devices. [8]

Both platforms use a combination of camera and sensor data to detect and track the user's position and orientation in the real world, and then use this data to place virtual objects into the user's real-world environment in a way that appears realistic and seamless. To achieve this, ARKit and ARCore use a variety of computer vision techniques, including feature detection, tracking, and scene understanding. For example, they may use machine learning algorithms to recognize and track the user's physical features and movements in real-time. In the case of the AR Try-On app by Shopify, the app uses ARKit and ARCore to place 3D models of products into the user's real-world environment, allowing them to visualize how the product would look in their space before making a purchase. The app uses machine learning algorithms to adjust the virtual product's position and orientation in response to the user's movements. [8]

I've had previous experience with ARCore and ArKit although they might seem like a good option for my app, they do not have pose/body tracking capabilities meaning they would not be able to predict points on they body which would be crucial for my app.



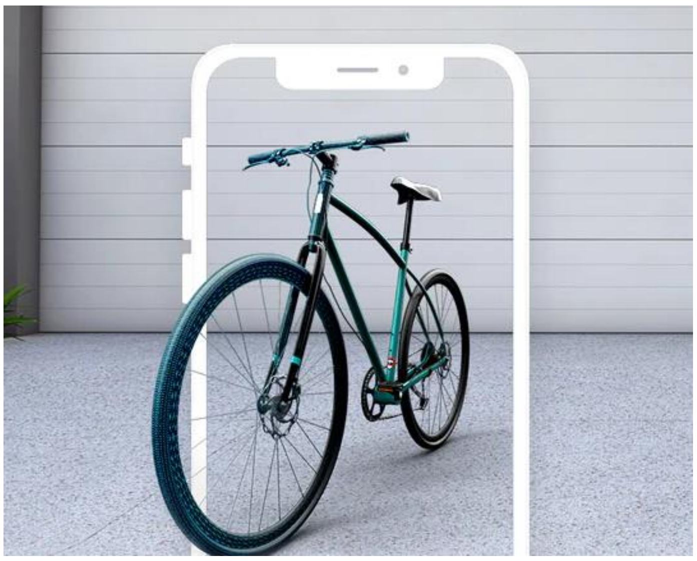


Fig.6 Video demonstration of Shopifys app link to the video - Introducing Shopify AR: Augmented Reality Shopping for Your Store [8]

Fig.7. Screenshot form Shopify website demonstration their app [8]

2.3 Metail

Metail is a virtual try-on app that uses 3D scanning and machine learning technologies to create a 3D avatar of the customer. The app then allows customers to try on clothes virtually and see how they would look and fit on their 3D avatar. Here's a brief overview of how Metail works:

The customer uploads two full-length photos of themselves, taken from the front and side, to the Metail app. Metail's 3D scanning technology analyses the photos to create a personalized 3D avatar of the customer, including their body shape, measurements, and proportions. The customer can then browse and select clothes from the app's virtual wardrobe, which features a wide range of clothing items from various brands. The selected clothing item is adjusted to fit the customer's 3D avatar in real-time using Metail's machine learning algorithms, which take into account factors such as fabric stretch and drape. The customer can view the clothing item on their 3D avatar from different angles, allowing them to see how it would look and fit on them.[9]

Fig.8 Video demonstration of Metails app [10]

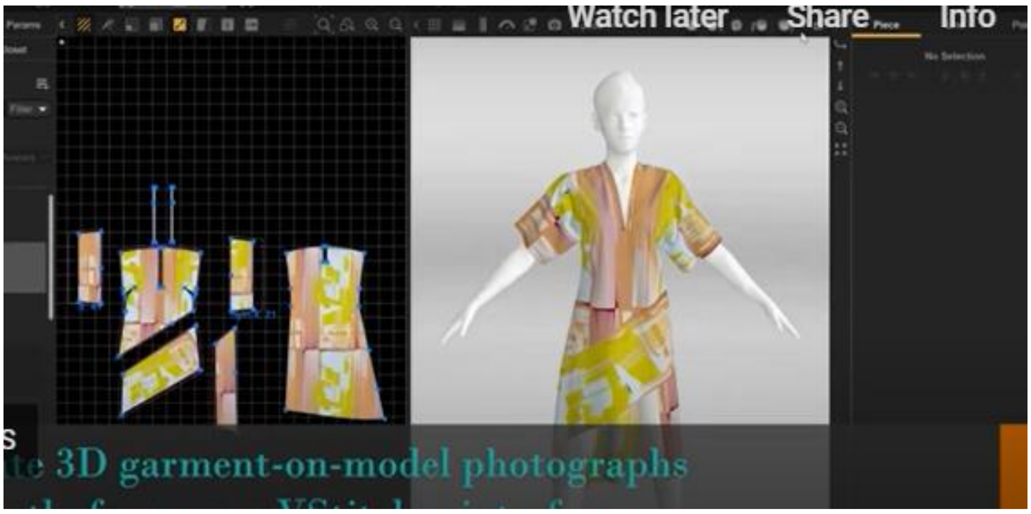


Fig.9 Image Showcasing Metails app [10]

2.4 Zeekit

Similar app to Metail is Zeekit which is a virtual fitting room app that uses computer vision and augmented reality technologies to allow customers to try on clothes. The Zeekit platform uses computer vision and deep learning algorithms to analyse a user's image and create a 3D virtual model of their body. This model can then be used to simulate how different articles of clothing would fit and look on the user, allowing them to try on clothes virtually before making a purchase. To use Zeekit, a user uploads a full-body image of themselves to the platform. They can then browse through the selection of clothes available on the platform or upload their own images of clothes they want to try on. When they select a garment, Zeekit's algorithms analyse the user's body shape and size and create a 3D model of the garment that fits the user's body. The user can then see how the garment looks on their virtual model from different angles and adjust the fit and size as necessary. Zeekit's technology is based on a combination of computer vision, machine learning, and AR. The company's computer vision algorithms analyze the user's image and create a 3D model of their body, which is then used to simulate the fit of different garments. The platform's machine learning algorithms continually learn and improve the accuracy of the simulations based on user feedback and data. [11]

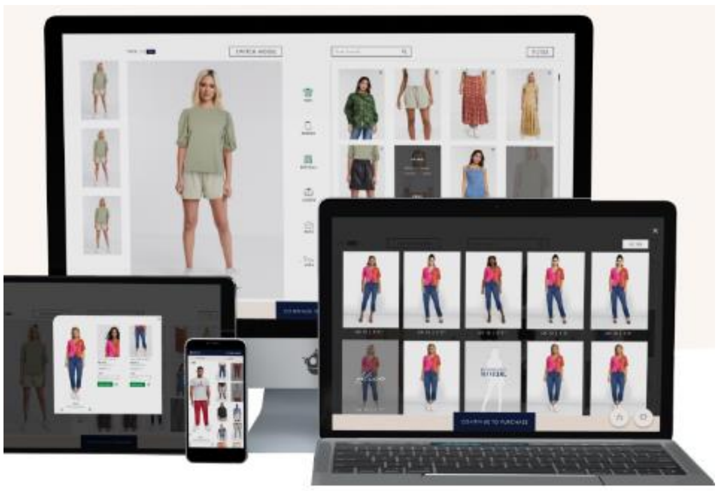


Fig.10 Image depicting Zeekits app. [11]

Fig.11 GIF depicitng Zeekits augmented reailty feature [11]

From my assessment I found that there are three main apps which are most similar to mine, which are Metail, Zeekit and YourFit. I will be comparing my finished product to these apps in a later segment of my report so that I can evaluate my app fairly.

3. Environments and Components

3.1 Unreal Engine

I have decided to develop my project using Unreal Engine.
Unreal Engine is a powerful game engine and development platform that is often used for creating interactive graphics and 3D environments. One of the benefits of using Unreal Engine for developing UI for apps is its high-quality graphics and visualization capabilities. The engine is equipped with advanced rendering technology, which allows developers to create visually stunning and immersive user interfaces. In addition, Unreal Engine's modular design and extensive set of tools make it easy for developers to customize and fine-tune the UI to meet their specific needs. In terms of AR, Unreal Engine has features that make it a great choice for creating AR applications. One of its main strengths is its ability to handle large amounts of 3D data and render it in real-time, which is essential for creating AR experiences that are interactive and immersive. Unreal Engine also includes built-in support for AR platforms, making it easy to develop AR applications for mobile devices. It also includes support for popular AR development frameworks like Vuforia and Wikitude. Another advantage of Unreal Engine for AR is its Blueprint visual scripting system, which allows developers to create complex AR experiences without needing to write code. This makes it easier for designers and artists to create AR content, as they can use Blueprint to create interactive elements, animations, and other visual effects. Overall, Unreal Engine is a popular choice for developing UI for apps due to its high-quality graphics, customization options, AR support and ease of use.

3.2 MediaPipe

Mediapipe is a powerful framework for building applications that require real-time data processing of media streams such as video and audio. It offers a set of pre-built and customizable pipelines that can be used for various tasks such as object detection, hand tracking, pose estimation, face recognition, and more. These pipelines use machine learning algorithms and computer vision techniques to process the media data and extract useful information. Mediapipe is particularly suitable for augmented reality development because it provides the necessary tools for tracking and mapping objects in real-time. Augmented reality (AR) is a technology that overlays virtual objects on top of the real world, often using a camera and computer vision algorithms to recognize and track objects. This requires high-performance and accurate tracking of objects in real-time, which can be achieved using Mediapipe's pre-built pipelines.

For example, Mediapipe's hand tracking pipeline can be used to track the user's hand movements in real-time, allowing for a more immersive AR experience where virtual objects can interact with the user's hands. The face detection and recognition pipelines can be used to overlay virtual makeup or accessories on a user's face, and the pose estimation pipeline can be used to track the user's body movements. Mediapipe's flexibility and ease of use make it a popular choice among developers for AR applications including the following.

AR Stickers - Google's AR Stickers app for Pixel and Pixel 2 devices uses MediaPipe to track the user's face and body movements and animate the AR stickers accordingly. The app also uses MediaPipe for hand tracking, allowing users to interact with the AR stickers using hand gestures.

AR Makeup - L'Oreal's Makeup Genius app uses MediaPipe for real-time facial tracking and analysis, allowing users to try on virtual makeup and see how it looks on their face in real-time.

AR Measurement - Google's Measure app uses MediaPipe for real-time object tracking and measurement, allowing users to measure the length, width, and height of objects in the real world using their smartphone camera.

AR Filters - Snapchat uses MediaPipe for real-time face and body tracking, allowing users to apply AR filters to their selfies and videos that track their movements and expressions.AR Navigation - Google's Live View AR navigation feature in Google Maps uses MediaPipe for real-time localization and orientation, allowing users to navigate through the real-world using AR overlays that provide directional cues and landmarks.

Here is a detailed breakdown on how Mediapipe works:

From the image below you can see our input goes through flow control which the ability to control the flow of data through the MediaPipe graph, which is a visual representation of the processing pipeline for a MediaPipe application. Flow control can be achieved through the use of MediaPipe's built-in flow control nodes, such as the Wait Node, which pauses the flow of data until a certain condition is met, or the Merge Node, which combines multiple data streams into a single output stream. After the data has been combined it gets pre-processed. Pre-processing refers to the manipulation of input data before it is fed into a machine learning model. In MediaPipe, pre-processing can be accomplished using Fig 20. Screenshot depicting Limitations of Augmented Reality aspect the framework's built-in pre-processing nodes, which can perform tasks such as image resizing, colour space conversion, and data normalization. Furthermore, it gets fed through model interference . Model inference refers to the process of using a trained machine learning model to make predictions on new input data. In MediaPipe, model inference can be performed using the TensorFlow Lite Interpreter API, which allows users to load and run pre-trained TensorFlow Lite models within the MediaPipe pipeline. After Model interference post processing takes place .Postprocessing refers to the manipulation of model output data after it has been generated by a machine learning model. In MediaPipe, postprocessing can be accomplished using the framework's built-in postprocessing nodes, which can perform tasks such as nonmaximum suppression, confidence thresholding, and data visualization. Once Post processing has been finished its time to synchronize. Synchronization refers to the coordination of data streams within the MediaPipe pipeline to ensure that different processing stages are operating in a coordinated and synchronized manner. In MediaPipe, synchronization can be achieved using the framework's built-in synchronization nodes, which can perform tasks such as waiting for data from multiple sources before processing can continue or ensuring that data streams are aligned in time with each other. [12]

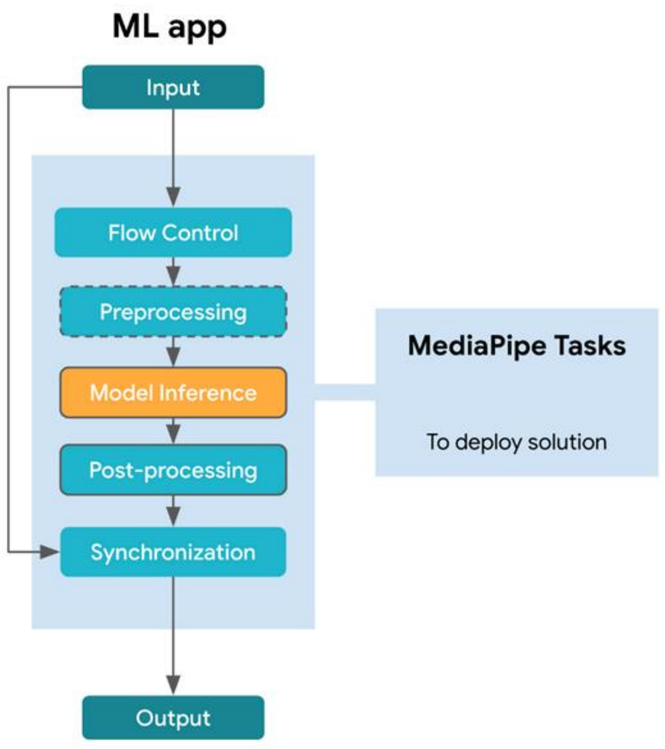


Fig.12 Medipipes framework visulization [12]

Mediapipe supports multiple platforms and programming languages, including Android, iOS, and TensorFlow. Its pre-built pipelines can be easily customized to fit specific use cases, and new pipelines can be created from scratch using Mediapipe's modular and extensible architecture.

Overall, Mediapipe's powerful machine learning algorithms and computer vision techniques make it a suitable framework for building high-performance and visually pleasing augmented reality applications. [13]

3.3 Firebase

Firebase is a comprehensive mobile and web development platform that provides a range of tools and services for building, scaling, and managing applications. One of the key features of Firebase is its powerful database and user authentication capabilities.

Firebase offers a flexible and scalable NoSQL cloud database that allows developers to store and sync data in real-time. This makes it an excellent choice for applications that require fast and reliable access to data, such as social networking, gaming, and messaging apps. In addition, Firebase's offline support allows users to access data even when they are not connected to the internet, making it a robust choice for building reliable and resilient applications.

Firebase also provides a range of user authentication options, including email and password, phone number, and third-party providers such as Google and Facebook. This makes it easy for developers to build applications that require user authentication and ensure that only authorized users can access certain features or data. Overall, Firebase is a comprehensive and reliable platform for building and managing applications. Its powerful database and user authentication capabilities make it an excellent choice.

3.4Wireframe

3.4.1) Log in Page and Register Page

After having decided what environments to use I began making prototypes of possible UI elements. Firstly, I started with the Login and Register page. The idea behind these was to give the user the option to register either as a User or a Designer that way once they have logged in, they would be treated with different UI options.

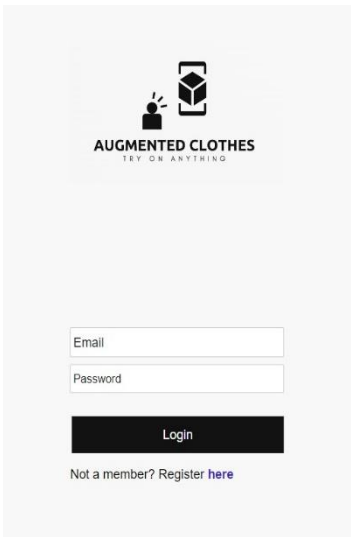


Fig 13. Initial Design of Login

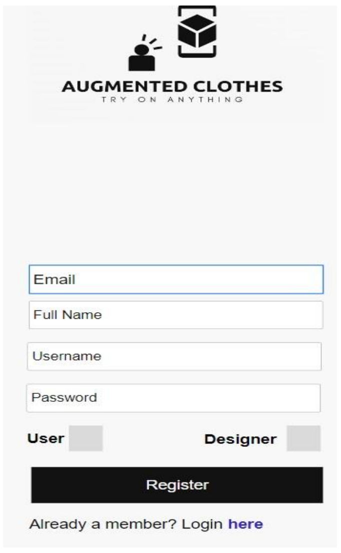


Fig 14. Initial Design of Sign-Up Page

3.4.2) Different User Views

Once a user signs up and logs in they are treated with the user main menu. In this view they have an option to Search outfits by various designers or by the name of the clothing. They would also have a saved outfits section and a popular section which showcases a list of popular designs.

Within the designer view , a Designer would have an option to view and edit their designs and upload new designs.

After a user selects a design from either saved outfits or popular tab the product view would appear, in this the user is presented with full size image of the product and the option to try it on.

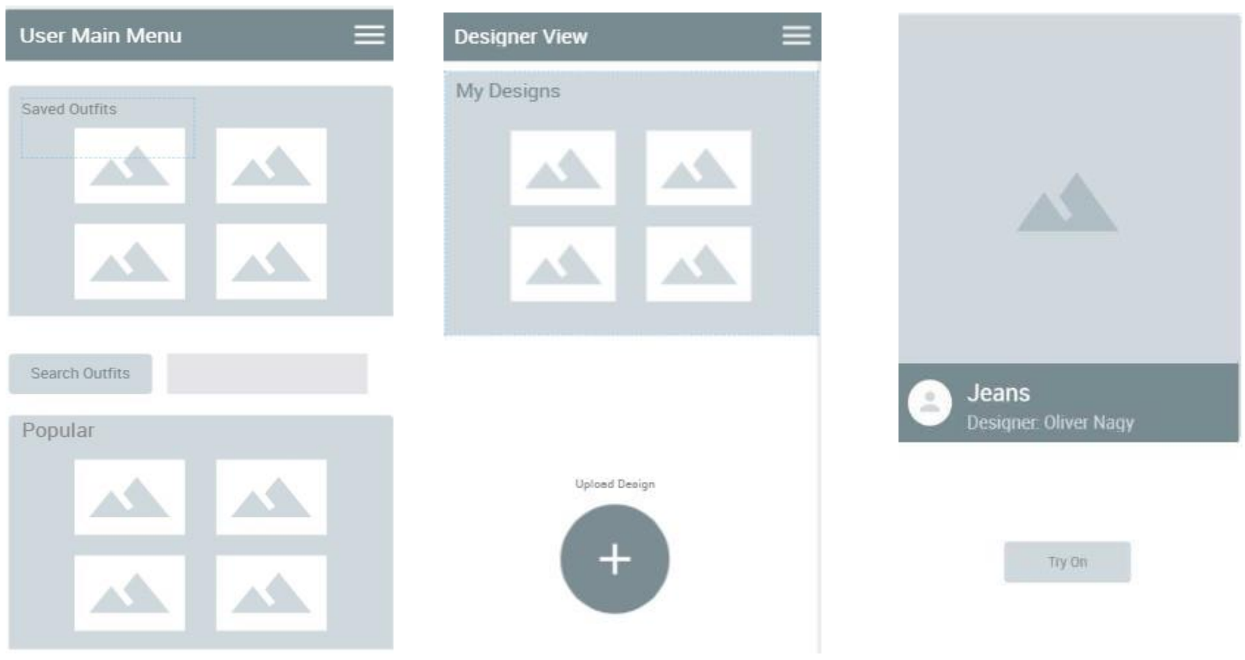


Fig 15. Initial Design of Main Menus Designer Menu and Product View

4. *Development*

**All code can be found in my final submission within the folders Assets -> Scripts . All Scripts within that folder were written entirely by me.*

4.1 Getting Started

I thoroughly read the documentation and gaining a thorough understanding of how MediaPipe works, I decided to create blueprints (C++ classes in Unreal) in order to integrate it into my project. These blueprints were then attached to objects within my scene, such as a 3D cube, in order to initiate the specified operation.

One example of this integration is the use of MediaPipe for 3D face tracking. This involves using the software to track the user's face and map points onto it, which can then be used to track the movement of the face. This functionality could be useful for applying virtual accessories, such as sunglasses, to the user's face in real-time. I was able to track the points and display them in my scene within a cube, as seen in the example on the left.

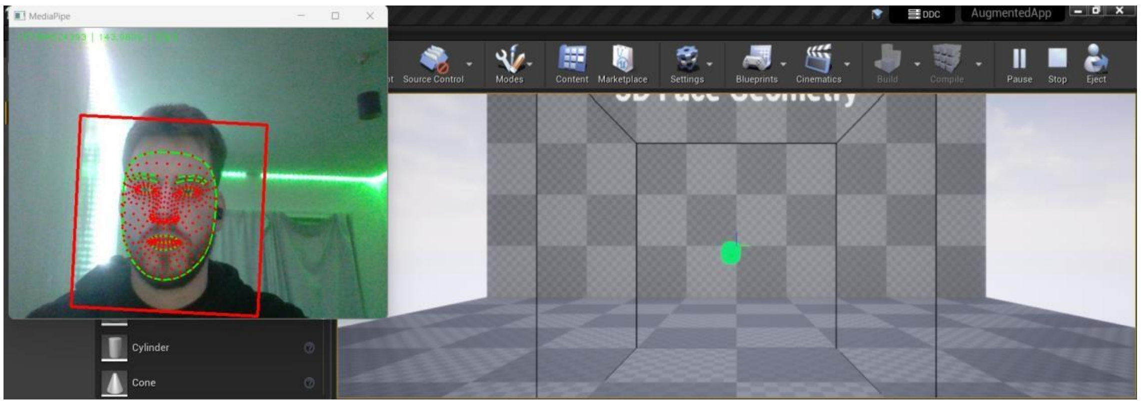


Fig 16. Screenshot of face tracking in Unreal

4.1.1) 3D Pose and Face tracking.

In this example, I combined two different blueprints - one for 3D face tracking and one for 3D pose tracking - to create an object or asset that is capable of tracking every part of the body in 3D. This allows for a more comprehensive and detailed tracking experience, covering not just the face but also the entire body. I then displayed some of the tracked points in my scene to demonstrate the capabilities of this combined tracking system. This type of comprehensive tracking could potentially be used for a variety of applications, not just for my AR app.

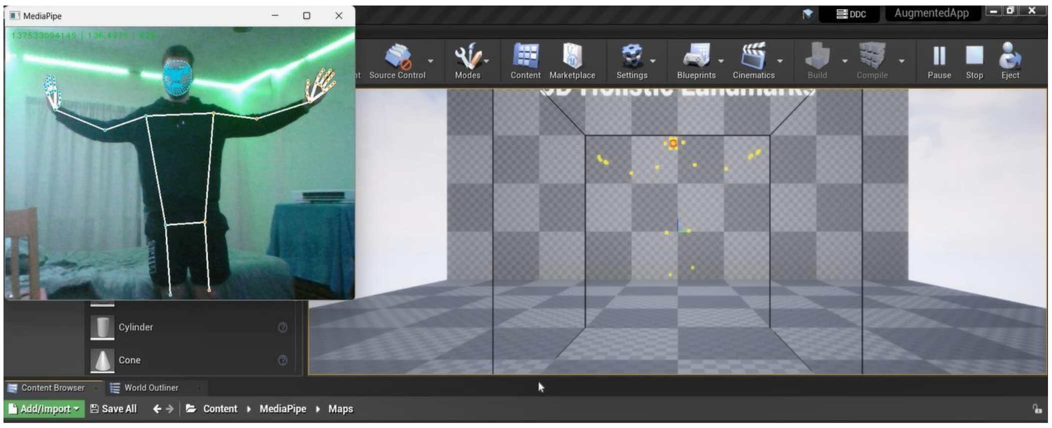


Fig 17. Screenshot of Holistic tracking in Unreal

4.1.2) 3D Pose tracking with Puppet.

In this example I used the blueprint developed in previous section above and attached it to a 3D puppet to showcase how it could potentially track any pieces of clothing in an augmented reality setting.

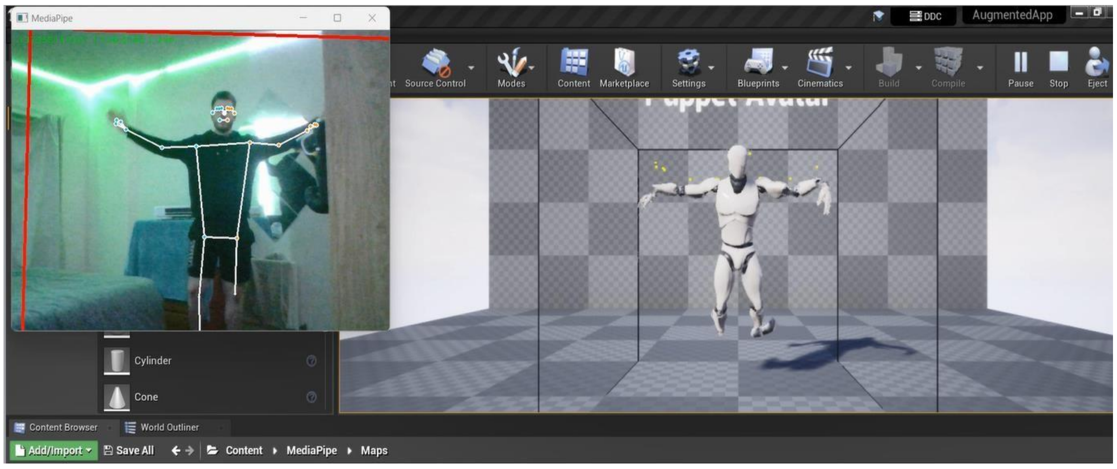


Fig 18. Screenshot of manipulating a puppet using Holistic tracking

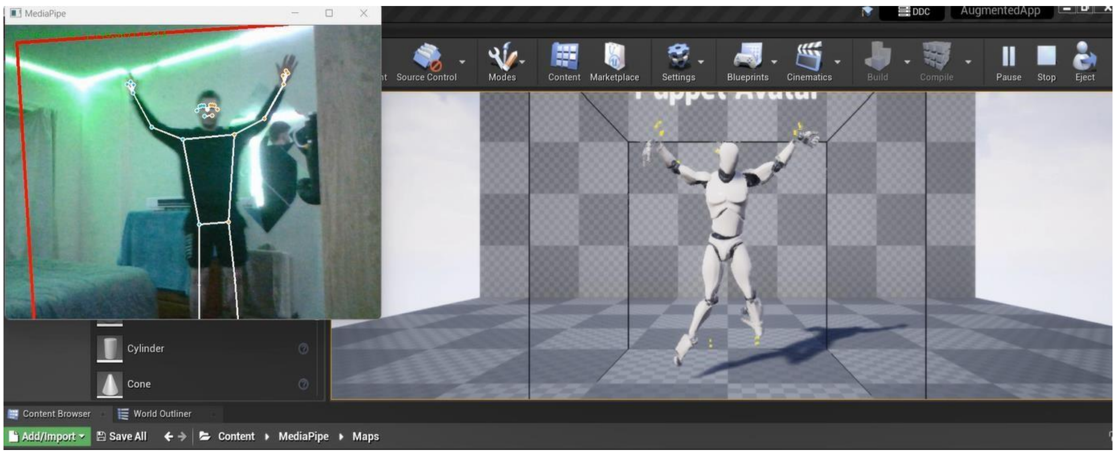


Fig 19. Screenshot of manipulating a puppet using Holistic tracking

As its 3D tracking it not only keeps track of points but, rotation of the points and which way they are facing therefore you can see once I turned around so did the puppet.

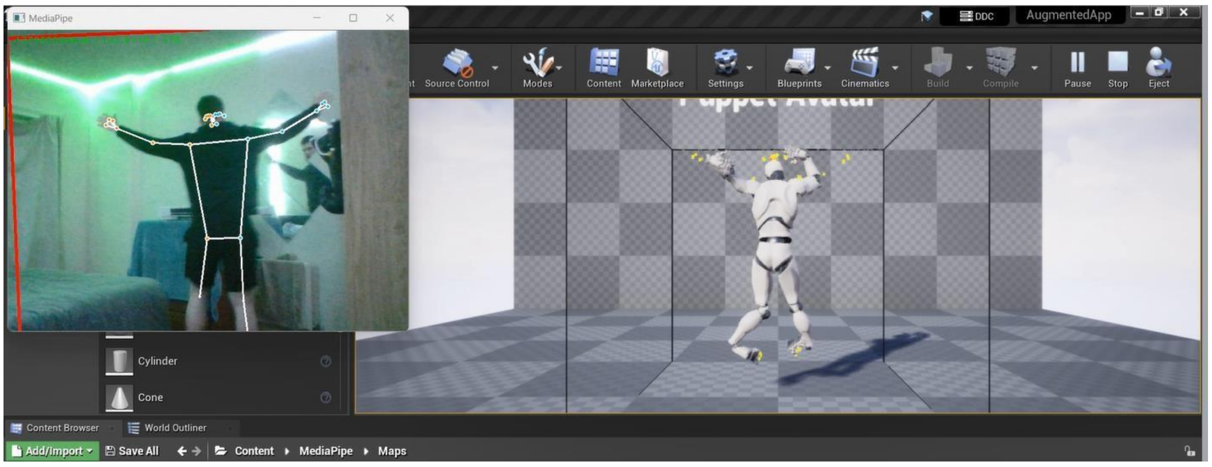
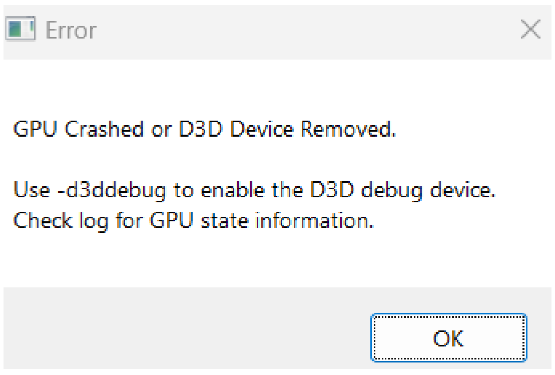


Fig 20. Screenshot of manipulating a puppet using Holistic tracking

4.2 Re-evaluation of Game Engine

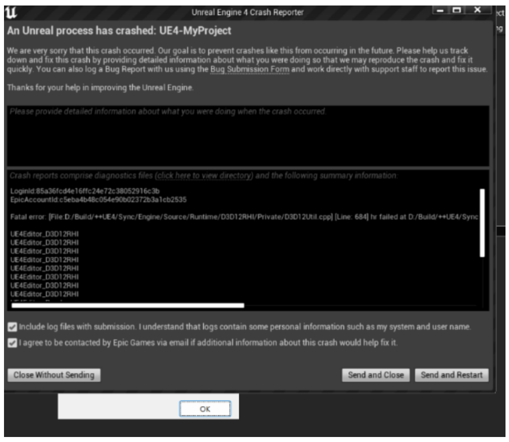
After having started my project in Ureal Engine , I came across some issues with Unreal Engine which had halted my development process. After thoroughly investigating the issue there seemed to be a problem with Unreal Engine and its Compatibility with my hardware specifically my GPU and drivers. After installing and re-installing Unreal Engine and updating the BIOS and Firmware on my GPU the problem couldn't be fixed. Therefore, I had a tough decision to make , to try and sink more time into trying to fix the issue , or the adapt and overcome this hurdle. After consideration I decided to try and re-create what I had already done on Unreal in Unity.



Error

GPU Crashed or D3D Device Removed.

Use -d3ddebug to enable the D3D debug device.
Check log for GPU state information.



An Unreal process has crashed: UE4-MyProject

We are very sorry that this crash occurred. Our goal is to prevent crashes like this from occuring in the future. Please help us track down and fix this crash by providing detailed information about what you were doing so that we may reproduce the crash and fix it quickly. You can also log a Bug Report with us using the Bug Submission Form and work directly with support staff to report this issue.

Thanks for your help in improving the Unreal Engine.

Please provide detailed information about what you were doing when the crash occurred

Crash reports comprise diagnostics files (click here to view directory) and the following summary information:

```
LoginId:85a36fcd4e16ffc24e72c38052916c3b
EpicAccountId:c5eba4b48c054e90b02372b3a1cb2535

Fatal error: [File:D:/Build/++UE4/Sync/Engine/Source/Runtime/D3D12RHI/Private/D3D12Util.cpp] [Line: 684] hr failed at D:/Build/++UE4/Sync

UE4Editor_D3D12RHI
UE4Editor_D3D12RHI
UE4Editor_D3D12RHI
UE4Editor_D3D12RHI
UE4Editor_D3D12RHI
UE4Editor_D3D12RHI
UE4Editor_D3D12RHI
UE4Editor_D3D12RHI
```

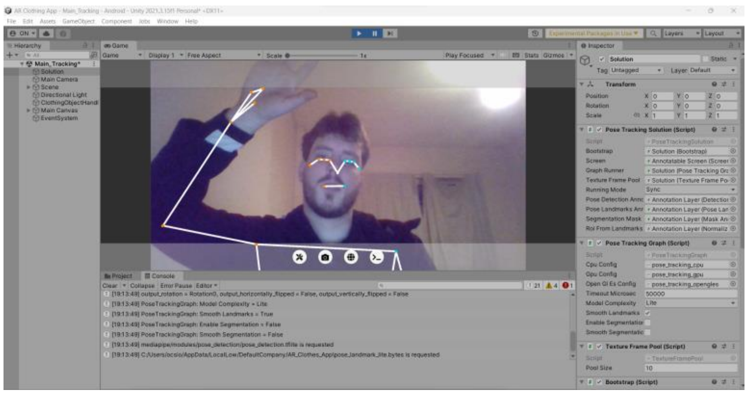
☑ Include log files with submission. I understand that logs contain some personal information such as my system and user name.

☑ I agree to be contacted by Epic Games via email if additional information about this crash would help fix it.

Fig 21. Screenshots of Unreal Engine errors

4.3 Setting up the environment in Unity

After I had to change game engines, I had got started on importing the Mediapipe plugin for Unity and setting up the project so that I could continue development. After reading through documentation and forums which can be found here - https://github.com/homuler/MediaPipeUnityPlugin/wiki/Installation-Guide , I was left with a functional pose tracking system and you can see below.

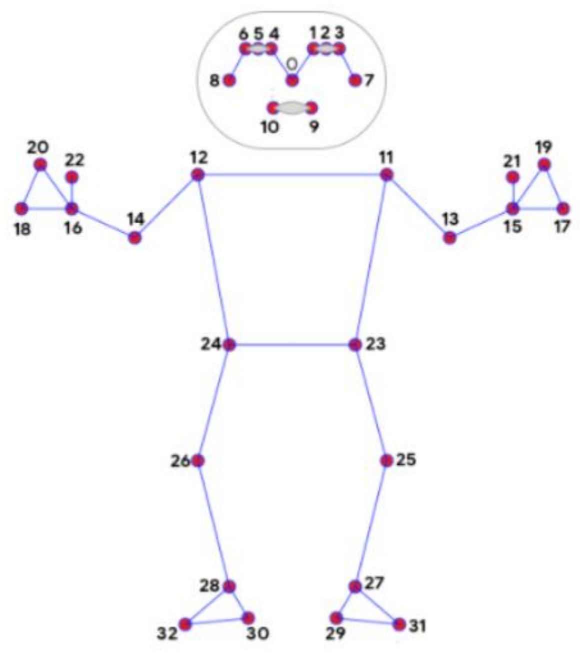


```
[19:13:48] output_rotation = Rotation0, output_horizontally_flipped = False, output_vertically_flipped = False
[19:13:48] PoseTrackingGraph: Model Complexity = Lite
[19:13:49] PoseTrackingGraph: Smooth Landmarks = True
[19:13:49] PoseTrackingGraph: Enable Segmentation = False
[19:13:49] PoseTrackingGraph: Smooth Segmentation = False
[19:13:49] mediapipe/modules/pose_detection/pose_detection.tflite is requested
[19:13:48] C:/Users/ocsio/AppData/LocalLow/DefaultCompany/AR_Clothes_App/pose_landmark_lite.bytes is requested
```

Fig 22. Screenshot of Unity environment after initial setup of Mediapipe

4.4 3D model

At the initial stages of development, I decided that I would implement a 3D model of a Shirt that would be applied to the body, at the time I thought this would be the best idea as I was going for the most realistic experience I could get. So, I started writing my own script Called ClothingHandler.cs . Before I could start writing the script, I had to make a few adjustments in the Media Pipe code. My main concern was that the holistic points that are being tracked , they don’t have an index in my project yet, so there was no way of telling which point was which during runtime which was a huge problem as I was planning to map the Clothing points to each “Landmark” or tracked point. My idea was to give each tracked points an index just like in the official MediaPipe Documentation as it shown below.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Fig 23. Image of Body points being tracked during Holistic tracking [13]

To achieve this, I figured out I had to edit two scrips called “Point List Annotation” and “Connection List Annotation” . Point List Annotation was responsible for displaying the tracked point on the Canvas / Screen and Connection List Annotation was responsible for Connection the points so that they would create a skeleton as shown above.

Both scripts have a method called “InstantiateChild” as shown below.

```
protected override ConnectionAnnotation InstantiateChild(bool isActive = true)
    {
      var annotation = base.InstantiateChild(isActive);
```

```
        annotation.SetColor(_color);
        annotation.SetLineWidth(_lineWidth);
        // responsible for name
        annotation.gameObject.name = _annotationPrefix + children.Count;
        return annotation;
    }
```

After finding that all I had to do was re-name the annotation based on their index which was just adding their number or “count” to the end of their name.

Now that I had labelled points it was easy enough to find them during runtime.

So now all I had to do was create the appropriate game objects within the script and then assign them to the tracked points.

```
 // Screen Annotation Points

    public GameObject ScreenRightShoulder;
    public GameObject ScreenLeftShoulder;
    public GameObject ScreenRightHip;
    public GameObject ScreenLeftHip;
    public GameObject ScreenRightWrist;
    public GameObject ScreenLeftWrist;
    public GameObject ScreenRightElbow;
    public GameObject ScreenLeftElbow;
```

As you can see above those are the points I’m going to be working with so that I can track the upper body and overlay my 3D shirt.

To assign each Game Object the appropriate point at runtime I decided to make separate method within my script called Parent Connection()

```
public void ParentConnection()
{

    // Assigning Screen Points

    ScreenRightShoulder = GameObject.Find("PointAnnotationScreen_11");
    ScreenLeftShoulder = GameObject.Find("PointAnnotationScreen_10");
    ScreenRightHip = GameObject.Find("PointAnnotationScreen_23");
    ScreenLeftHip = GameObject.Find("PointAnnotationScreen_22");
    ScreenRightWrist = GameObject.Find("PointAnnotationScreen_15");
    ScreenLeftWrist  = GameObject.Find("PointAnnotationScreen_14");
    ScreenRightElbow = GameObject.Find("PointAnnotationScreen_13");
    ScreenLeftElbow = GameObject.Find("PointAnnotationScreen_12");
```

which would be called in the Awake() method. The Awake method is one of the built in methods that come with Unity . It runs before the script starts. I had done this because when I start the app we first need to wait for the tracking to become active and start tracking points, If we try and assign a Point Annotation to a Game Object before the annotation had been created we would get a Null Reference error. So this was my way of synchronising the whole operation.

```
    void Awake()
    {
        Shirt.SetActive(false);
        Invoke("ParentConnection", ScriptStartUp);

    }
```

The awake method starts at the same time as we run the app, After which it Invokes the ParentConnection method with a time delay called ScriptStartUp which is set to 1.5 second this gives the tracking system enough time to track all the points therefore avoiding a null reference error.

After I had the tracked points assigned it was time to apply a 3D model of a Shirt to it to test it out. I decided to get my model from TurboSquid.com which is a royalty free site where you can download model.

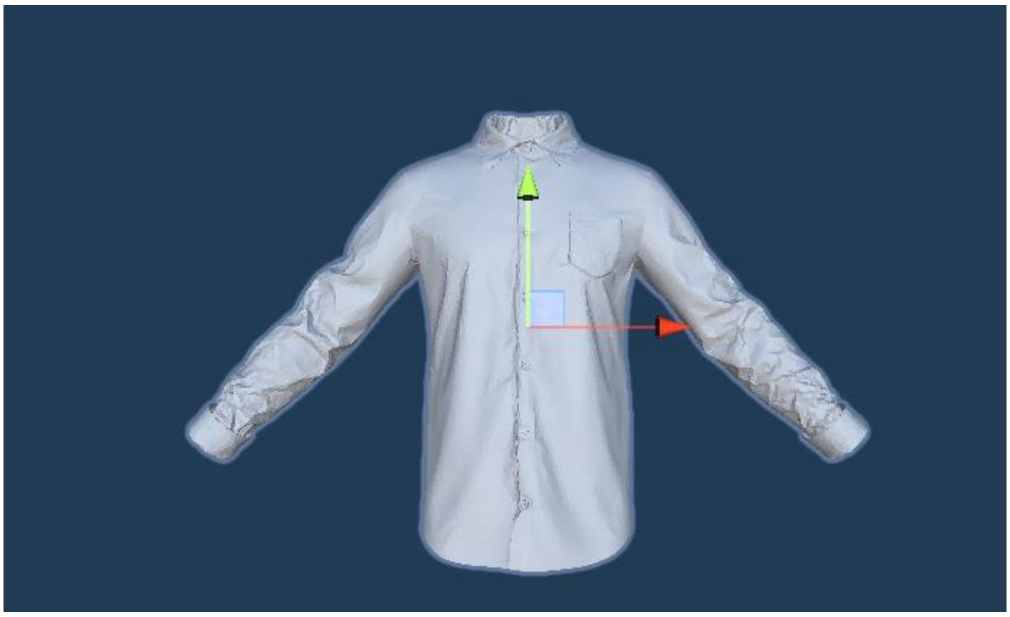


Fig 24. Image showing 3D model of a shirt.

Then I started writing the ShirtHandler() method which would be called in the Update() method. The update method is another built in method that comes with unity, It runs every frame therefore I can be sure that whatever calculations I perform would be applied instantly.

```
 // Calculate the midpoint between ScreenLeftShoulder and ScreenRightShoulder
    Vector3 midpointShoulders = (ScreenLeftShoulder.transform.position +
ScreenRightShoulder.transform.position) / 2f;

    // Calculate the midpoint between ScreenLeftHip and ScreenRightHip
    Vector3 midpointHips = (ScreenLeftHip.transform.position +
ScreenRightHip.transform.position) / 2f;

    // Calculate the direction of the line from midpointShoulders to midpointHips
    Vector3 lineDirection = midpointHips - midpointShoulders;

    // Calculate the midpoint of the line between midpointShoulders and
midpointHips
    Vector3 midpointLine = midpointShoulders + lineDirection / 2f;

    // Draw a line from midpointShoulders to midpointLine
    Debug.DrawLine(midpointShoulders, midpointLine, Color.green);

    // Draw a line from midpointLine to midpointHips
    Debug.DrawLine(midpointLine, midpointHips, Color.green);

    // Set the position of the Shirt GameObject to the midpoint of the line
between midpointShoulders and midpointHips
    Shirt.transform.position = midpointLine;
```

Above you can see the calculation that I came up with to place the shirt in the middle of the body, as this was called every second whenever you moved the shirt moved with you.

This worked great but I realized there was a problem. As I moved towards and away from the screen the Shirt was not scaling , or in other words it was staying a constant size. This was a big problem as you want the shirt appear smaller the further you are away from the screen and vice versa. So I decided to scale the Shirt as shown below.

```
  // Scale the Shirt GameObject based on the average distance between
ScreenRightShoulder and ScreenRightHip and between ScreenLeftShoulder and
ScreenLeftHip
    float distance = Vector3.Distance(ScreenRightShoulder.transform.position,
ScreenRightHip.transform.position) +
Vector3.Distance(ScreenLeftShoulder.transform.position,
ScreenLeftHip.transform.position);
    float scale = distance * clothingScale / 2f;
    Shirt.transform.localScale = new Vector3(scale, scale, scale);
```

The idea was that the further a person is from the screen the smaller the lines connecting the tracked points are. Therefore, I could have a scale which I can compare my Shirt to. I added the variable clothingScale to tune this scaling of the shirt.

This also worked great and my shirt was scaling down and up with my movement away from the screen.

Now it was time to move onto rotation of the Shirt. This was by far one of the hardest obstacles I faced so far. The challenge was that I had tracked point such as `ScreenRightShoulder` but as the name suggests these are only 2D point on a screen therefore the don't rotate or scale , in other terms these aren't World Points with x,y and z values or x and y. To solve this problem, I decided to make a set of 3D points within my scene but hidden from the user , as you can see below.

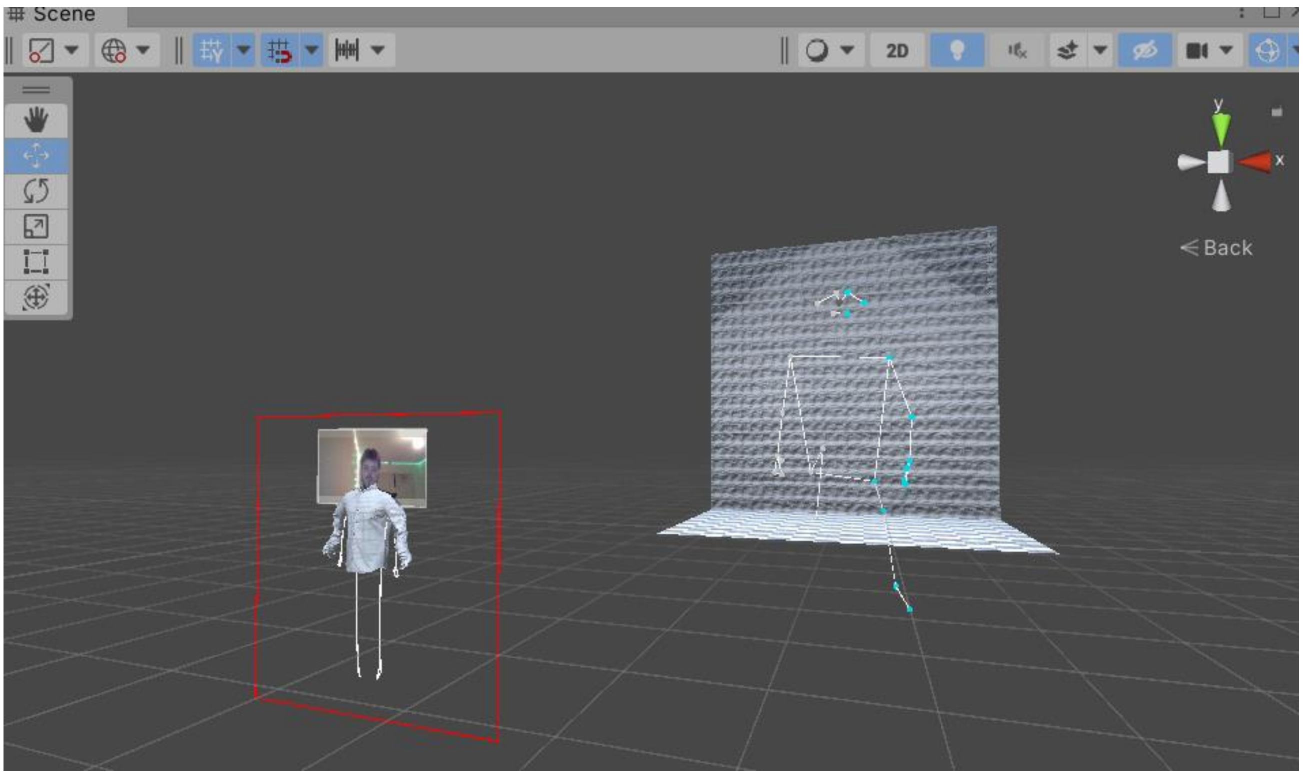


Fig 25. Screenshot showing the Unity environment. Screen Points are visible on the righthand side while World Points are visible on the lefthand side.

This second representation of the tracked points of the body are similar to the one on screen but if you have a look at the snippet below so can see that these are 3D dimensional with World Values.

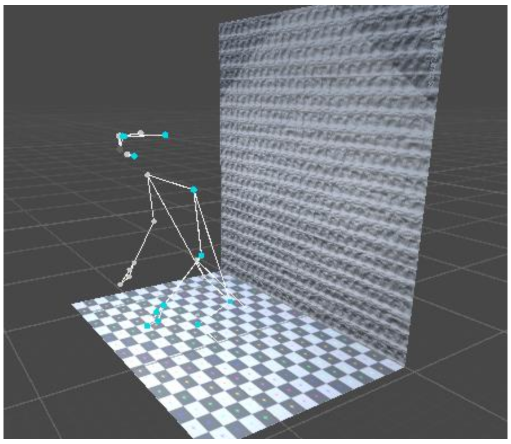


Fig 26. Screenshot showcasing visual representation of World Points

Now all that was left was to create a new set of tracked points called WorldPoints.

```
 //World Annoatation Points
    public GameObject WorldLeftShoulder;
    public GameObject WorldRightShoulder;
    public GameObject WorldRightHip;
    public GameObject WorldLeftHip;
    public GameObject WorldLeftElbow;
    public GameObject WorldRightElbow;
    public GameObject WorldLeftWrist;
    public GameObject WorldRightWrist;
```

Once this was done I also parented /assigned these Game Objects to the tracked points within the Parent Connection method.

Then I started working on the rotation. My initial idea was to put a cube around the four points that were the two hips and to shoulder then whenever you would move there would be some rotation applied to the cube ,this would all happen in the Scene at runtime and not be visible to the user. Then I could map this rotation to the Shirt and hopefully it would rotate as you would rotate your body. I also added a disable rotation Boolean for testing and debugging purposes.

```
if (!DisableRotation)
{
// Calculate the rotation of the cube formed by the four GameObjects
WorldLeftShoulder, WorldRightShoulder, WorldRightHip, and WorldLeftHip
Vector3 cubeForward = (WorldRightShoulder.transform.position -
WorldLeftShoulder.transform.position).normalized;
Vector3 cubeRight = (WorldRightHip.transform.position -
WorldRightShoulder.transform.position).normalized;
Vector3 cubeUp = Vector3.Cross(cubeForward, cubeRight).normalized;
Quaternion cubeRotation = Quaternion.LookRotation(cubeForward, cubeUp);
        // Check if the change in rotation of the cube exceeds the threshold
if (Quaternion.Angle(cubeRotation, prevCubeRotation) > threshold)
{
    // Align the Shirt GameObject's x, y, and z axes with the x, y, and z axes of
the cube
    Shirt.transform.rotation = cubeRotation;
}

// Apply the rotation of the cube to the Shirt GameObject
Shirt.transform.rotation *= cubeRotation;

// Update the previous cube rotation
prevCubeRotation = cubeRotation;
}
```

This had worked well , but the problem was the whole shirt was rotating and it wasn't anchored to the body.

To fix this I decided to try and anchor some points of the shirt to the body by just mapping them 1 to 1 as shown below.

```
// Map the rigged shirt to its corresponding points on the screen
    ShirtLeftShoulder.transform.position = ScreenLeftShoulder.transform.position;
    ShirtRightShoulder.transform.position =
ScreenRightShoulder.transform.position;
    ShirtHip.transform.position = midpointHips;
    ShirtLeftWrist.transform.position = ScreenLeftWrist.transform.position;
    ShirtRightWrist.transform.position = ScreenRightWrist.transform.position;
    ShirtLeftElbow.transform.position = ScreenLeftElbow.transform.position;
    ShirtRightElbow.transform.position = ScreenRightElbow.transform.position;
```

but this didn't work as it produced some weird effects due to the fact that if you map points 1 to 1 the mesh of the object gets distorted as you're not applying the rotation of the object just the position.

As you can see below the shirt maps somewhat well to the body but as soon as you move your arms it gets distorted

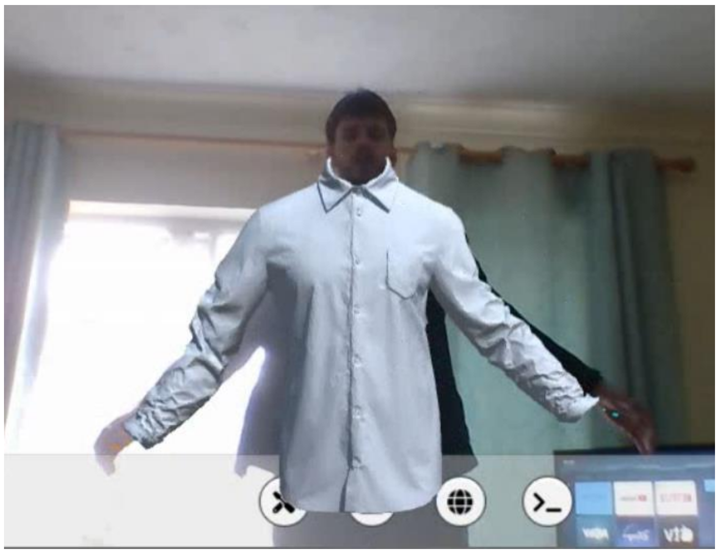


Fig 27. Screenshot showing limitations and distortions within 3D model .

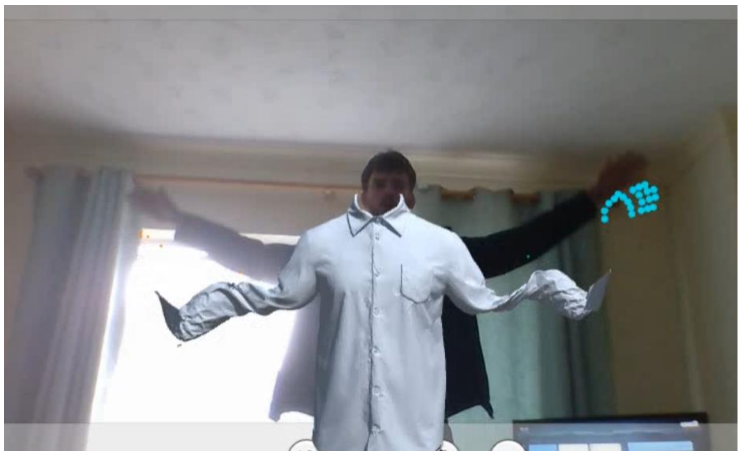


Fig 28. Screenshot showing limitations and distortions within 3D model .

To combat this, I knew that I couldn't just do what I had done previously and map 3D shapes like a cube to the World Points as for the arm there are not enough points. Therefore, I figured I had to use a combination of both the Screen Point and World Points. The difficulty with this was that they both have different values therefore how could I accurately combine them to provide me with a realistic representation of the position and rotation of the arms.

After doing some research I came across the RANSAC method.

"Random sample consensus, or RANSAC, is an iterative method for estimating a mathematical model from a data set that contains outliers. The RANSAC algorithm works by identifying the outliers in a data set and estimating the desired model using data that does not contain outliers".

„RANSAC is accomplished with the following steps

1. Randomly selecting a subset of the data set
2. Fitting a model to the selected subset
3. Determining the number of outliers
4. Repeating steps 1-3 for a prescribed number of iterations

For example, the equation of a line that best fits a set of points can be estimated using RANSAC."

[14]

The full implementation of the RANSAC method can be found under section 6. Its made up of three methods:

```
private Matrix4x4 ComputeTransform(List<Vector3> worldPoints, List<Vector3>
screenPoints)
```

```
private int[] RandomSample(int populationSize, int sampleSize)
```

```
public Matrix4x4 FindTransform(List<Vector3> worldPoints, List<Vector3>
screenPoints)
```

After implementing RANSAC I found that it didn't really work as planned , the bestTransform which it returns wasn't always correct and didn't map the shirt correctly during runtime. At this stage I was concerned that I was overcomplicating the implementation, which prompted me to re-evaluate my approach. During my studies, I came across the KISS design principle, which emphasizes simplicity in system design. By keeping systems as simple as possible, complexity is reduced, and user acceptance and interaction are maximized.

As I thought about my app's requirements, I realized that a user would not need to fully rotate to try on clothes, and therefore, only limited rotation of the model would be necessary. Consequently, a 2D model that only accommodates rotation in the x and y directions, rather than the z direction, would suffice. In addition to simplicity, other factors, such as ease of development and faster loading times, also influenced my decision to use a 2D image model. Another advantage of using a 2D image model for an augmented reality application is that it is easier to upload for users than a 3D object file. With a 2D image, users can quickly and easily take a picture or select an image from their device's library. In contrast, 3D objects files require more processing power and can take longer to upload, potentially resulting in slower loading times for the user. Overall, using a 2D image model for my augmented reality application offers several benefits, including simplicity, faster loading times, and ease of development.

Therefore, I decided to put a pause on the 3D model and with the knowledge I gained I decided to make one use only 2D models or images of shirts.

4.5 2D model

4.5.1) Long Sleeve Clothing ***– LongSleeveClothing.cs***

To get started I used a variation of ClothingHalder.cs I had written earlier the main difference being that I didn't use World Co-ordinates only Screen Co-ordinates.
I've started with position the Shirt in the centre of the body after which I applied scaling to it to match the persons body size, Furthermore rotation in the x and y axis based on the rotation of the line formed by the two hip points and the two shoulder points

Fig.29 Image of Long Sleeve clothing used for development.

```
  // Calculate the angle between the direction vector of the line between
ScreenLeftShoulder and ScreenRightShoulder and the x-axis
    float angle = Mathf.Atan2(lineDirectionShoulders.y, lineDirectionShoulders.x)
* Mathf.Rad2Deg;

    // Rotate the Shirt GameObject to match the direction vector
    Shirt.transform.rotation = Quaternion.Euler(0f, 0f, angle);
    // Scale the Shirt GameObject based on the average distance between
ScreenRightShoulder and ScreenRightHip and between ScreenLeftShoulder and
ScreenLeftHip
    float distance = Vector3.Distance(ScreenRightShoulder.transform.position,
ScreenRightHip.transform.position) +
Vector3.Distance(ScreenLeftShoulder.transform.position,
ScreenLeftHip.transform.position);
    float scale = distance * ShirtScale / 2f;
    Shirt.transform.localScale = new Vector3(scale, scale, scale);
```

After this was done, I repeated the steps for each individual part of the shirt e.g. left arm and right arm so that it maps to the body of the user.

```
//RightArm mapping
    Vector3 rightArmDirection = ScreenRightWrist.transform.position -
ScreenRightElbow.transform.position;


    ShirtRightArm.transform.position = ScreenRightElbow.transform.position;

    Vector3 rightArmEulerAngles = ShirtRightArm.transform.eulerAngles;
    rightArmEulerAngles.z = Mathf.Atan2(rightArmDirection.y, rightArmDirection.x)
* Mathf.Rad2Deg;
    ShirtRightArm.transform.eulerAngles = rightArmEulerAngles;
```

After I finished this I had a pretty well functioning 2D model of a Long Sleeve Shirt being mapped to the user's body. Therefore, I decided to re-name this script LongSleeveClothingHandler.cs . After this I had a methodology for mapping 2D images/sprites to the user which I was happy with therefore I decided to move onto different body parts and clothing.

4.5.2) Trousers ***– TrousersHandler.cs***

The next genre of clothing I decided to implement was Trousers. For this the methodology was very similar to Long Sleeve Clothing with the main exception being that I had to map the Trouser to different body points as you can see below.

```
  Vector3 hipMidPoint = (ScreenRightHip.transform.position +
ScreenLeftHip.transform.position) / 2;
        Vector3 kneeMidPoint = (ScreenRightKnee.transform.position +
ScreenLeftKnee.transform.position) / 2;
        yOffset = kneeMidPoint.y - hipMidPoint.y;
        // Place the TrouserRightLeg GameObject on the line that is formed by
ScreenRightHip and ScreenRightKnee
        TrouserRightLeg.transform.position =
Vector3.Lerp(ScreenRightHip.transform.position,
ScreenRightKnee.transform.position, 0.1f);
        // Place the TrouserRightShin GameObject on the line that is formed by
ScreenRightKnee and ScreenRightAnkle
        TrouserRightShin.transform.position =
Vector3.Lerp(ScreenRightKnee.transform.position,
ScreenRightAnkle.transform.position, 0.1f);

// Place the TrouserLeftLeg GameObject on the line that is formed by
ScreenLeftHip and ScreenLeftKnee
        TrouserLeftLeg.transform.position =
Vector3.Lerp(ScreenLeftHip.transform.position, ScreenLeftKnee.transform.position,
0.1f);
        // Place the TrouserLeftShin GameObject on the line that is formed by
ScreenLeftKnee and ScreenLeftAnkle
        TrouserLeftShin.transform.position =
Vector3.Lerp(ScreenLeftKnee.transform.position,
ScreenLeftAnkle.transform.position, 0.1f);
```

Fig.30 Image ofTrousers used for development.

4.5.3) Eye Wear ***-GlassesHandler.cs***

Implementing glasses / eye wear was somewhat similar to that previously mentioned above. The main difference being that I was only dealing with two points which was the right and left eye . First, I had to import the face and Iris tracking framework provided by MediaPipe. Then I had to calculate the mid-point between the left and right iris and the move the glasses there I also had to calculate the direction vector formed by the line formed by the left and right iris then I applied the rotation of that line to the Glasses and finally scaled the glasses to match the users face.

```
 // Calculate the midpoint between the left and right eye
    Vector3 midpoint = (ScreenLeftEye.transform.position +
ScreenRightEye.transform.position) / 2f;

    // Calculate the distance between the left and right eye
float distance = Vector3.Distance(ScreenLeftEye.transform.position,
ScreenRightEye.transform.position);

// Apply a y-offset to the midpoint position
midpoint.y += glasessyOffset;

// Move the glasses to the adjusted midpoint position
Glasses.transform.position = midpoint;

// Calculate the direction vector between the left and right eye
Vector3 direction = ScreenRightEye.transform.position -
ScreenLeftEye.transform.position;
direction.z = 90f; // set the z component to 0 to prevent rotation in the z
direction
direction.Normalize(); // normalize the direction vector

// Calculate the angle between the direction vector and the x-axis
float angle = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg;

// Rotate the glasses GameObject to match the direction vector
Glasses.transform.rotation = Quaternion.Euler(0f, 0f, angle);

  Glasses.transform.Rotate(0f, 0f, 180f);


// Scale the glasses based on the distance between the left and right eye
float scaleFactor = distance / 2f; // divide by 2 to scale proportionally
Glasses.transform.localScale = new Vector3(scaleFactor, scaleFactor, scaleFactor) *
GlassesScale;
```

Fig.31 Image of Eye Wear used for development.

4.5.4) Head Wear – ***HatHandler.cs***

Implementing Hats was similar to that to glasses with the major difference being that it also tracks the top of the head to position the Head Wear there.

```
 // Position the hat at the forehead top point
        Vector3 hatPos = ForheadTop.transform.position;
        hatPos.y += hatyOffset;
        Hat.transform.position = hatPos;

        // Calculate the direction vector from left to right eye
        Vector3 eyeDirection = ScreenRightEye.transform.position -
ScreenLeftEye.transform.position;

        // Calculate the direction vector between the left and right eye
        Vector3 direction = ScreenRightEye.transform.position -
ScreenLeftEye.transform.position;
        direction.z = 90f; // set the z component to 0 to prevent rotation in the z
direction
        direction.Normalize(); // normalize the direction vector

        // Calculate the angle between the direction vector and the x-axis
        float angle = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg;

        // Rotate the Hat GameObject to match the direction vector
        Hat.transform.rotation = Quaternion.Euler(0f, 0f, angle);
        Hat.transform.Rotate(0f, 0f, 180f);

        // Scale the hat based on the distance between the left and right eyes
        float distance = Vector3.Distance(ScreenLeftEye.transform.position,
ScreenRightEye.transform.position);
```

```
        Hat.transform.localScale = new Vector3(distance * HatScale, distance * 
HatScale, distance * HatScale);
```

With Finishing the Hat Handler, the development of the Augmented Reality aspect was finished. Therefore, it was time to move on to the User Interface.

Fig.32 Image of Head Wear used for development.

4.6 User Interface and Database

For the UI I used a lot of Unity's built in features such buttons and texts which made it a lot easier and faster to implement the UI. Therefore, I'm only going to talk about the Scripts I had written to control the UI and the Database behind keeping and storing users information.

4.6.1) Image Uploader ***– ImageUploader.cs***

This script is responsible for handling the uploading of images from the user's device and loading the images within the clothes Library (Library of all clothes that the user can try on). One of the notable things this script does it that it uploads the images and also automatically makes a button out of it so that way the user can press it and therefore select it. It also keeps track of which piece of clothing was selected last in each genre and that's what will be loaded in main scene.

4.6.2) Clothing Loader – ***ClothesLoader.cs***

This script is responsible for Loading the selected Clothes from the Clothes Library into the main scene. One notable feature of this script is that when it loads the images it sets their pivot and scale to match the previously defined size by me which ensures that there aren't any weird visual effects.

4.6.3) Sign In and Login - ***SingInHandler.cs & LoginHandler.cs***

These two scripts are responsible for signing a user up and logging them in. This is where Firebase discussed in chapter 3 section 3 comes into play. Connecting Firebase to my Unity project involved a few steps which are outlined below.

1. I created a new Firebase project by going to the Firebase console and clicking on "Add Project". I then followed the prompts to create a new project.
2. I added Firebase to my Unity project by following these steps:

a. I downloaded the Firebase Unity SDK from the Firebase website.

b. I imported the Firebase Unity SDK into my Unity project by going to the Assets menu, selecting Import Package, and then selecting Custom Package. I browsed to the location of the Firebase Unity SDK that I downloaded and selected it.

c. In the Firebase console, I selected my project and then clicked on "Add app". I followed the prompts to add an app to my project. I was prompted to enter a package name for my app, which I matched with the package name of my Unity project.

d. I downloaded the google-services.json file from the Firebase console and placed it in my Unity project's Assets folder.

3. I initialized Firebase in my Unity project by adding the following code to a script in my project:

```
using Firebase;
using Firebase.Unity.Editor;

public class FirebaseInitializer : MonoBehaviour {
    void Start() {
        FirebaseApp.DefaultInstance.SetEditorDatabaseUrl("MY_DATABASE_URL");
    }
}
```

I replaced "MY_DATABASE_URL" with the URL of my Firebase database, which I found in the Firebase console.

After the steps above I had connected Firebase with my Unity project which adds functionality to my app. I could use Firebase to store and retrieve data, authenticate users, and send notifications. I consulted the Firebase documentation for details on how to use these features.

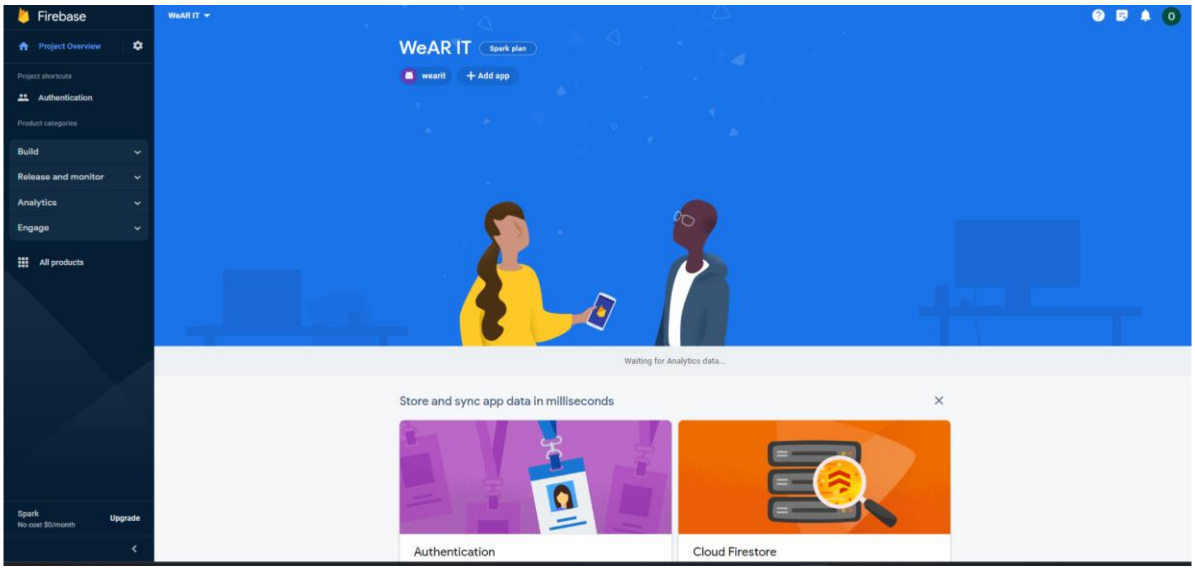
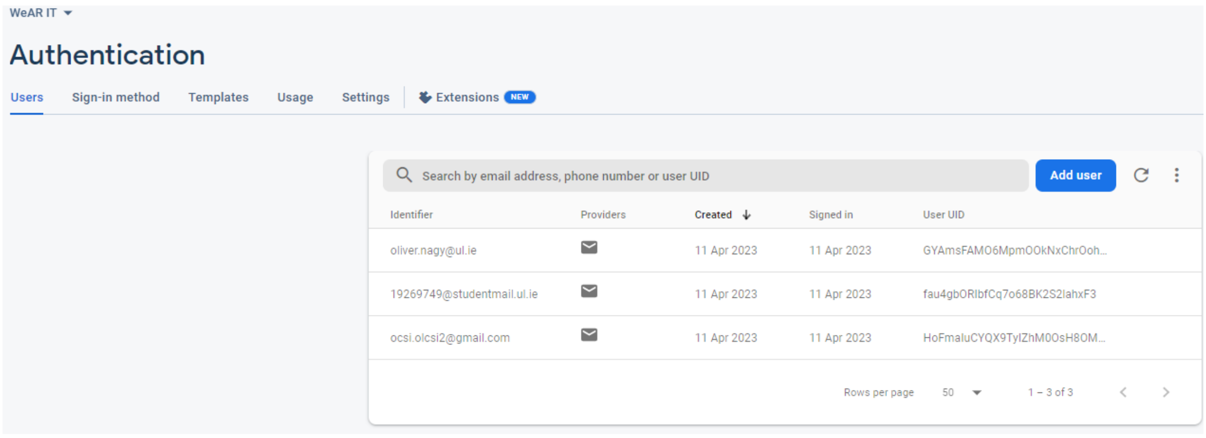


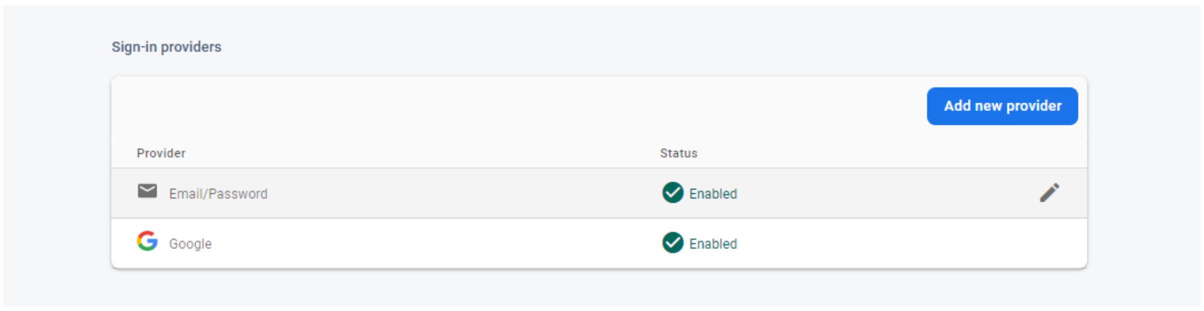
Fig 33. Screenshot showing Firebase interface for my project.



Identifier	Providers	Created	Signed in	User UID
oliver.nagy@ul.ie	✉	11 Apr 2023	11 Apr 2023	GYAmsFAMO6MpmOOkNxChrOoh...
19269749@studentmail.ul.ie	✉	11 Apr 2023	11 Apr 2023	fau4gbORlbfCq7o68BK2S2lahxF3
ocsi.olcsi2@gmail.com	✉	11 Apr 2023	11 Apr 2023	HoFmaluCYQX9TylZhM0OsH8OM...

Rows per page 50 1 – 3 of 3

Fig 34. Screenshot showing interface for managing users.



Provider	Status
Email/Password	Enabled
Google	Enabled

Fig 35. Screenshot showing interface for managing sign in providers.

5. *Final Product*

5.1 UI

I've managed to develop an easy to use and simple layout for my user interface with a universal theme and colour palette throughout the app. Below is the Login and Signup page.

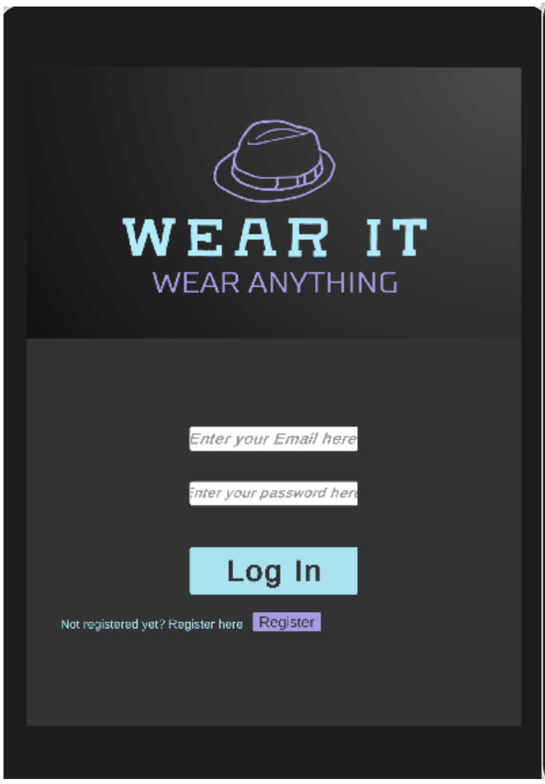


Fig 36. Screenshot showing finished UI for Login Page

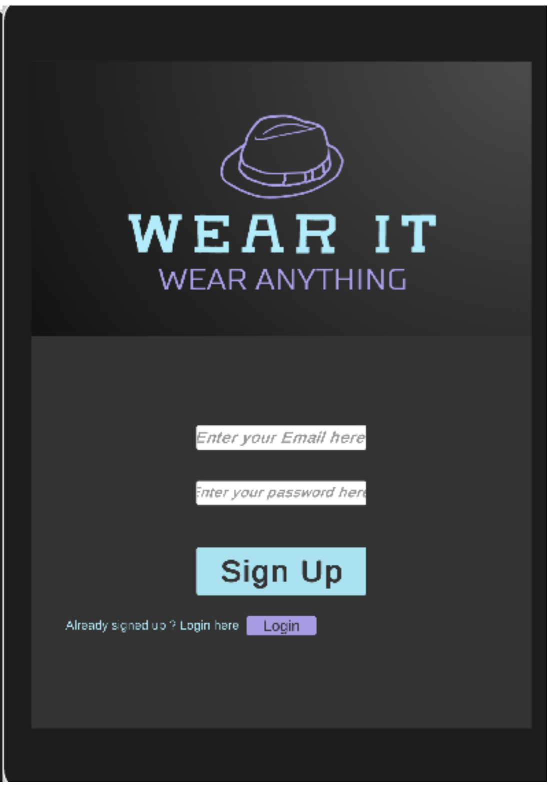


Fig 37. Screenshot showing finished UI for Signup Page

After the user logs in the can upload and Select Clothes from the clothes Library. This way once they have uploaded their choice of clothing they can always find it later.

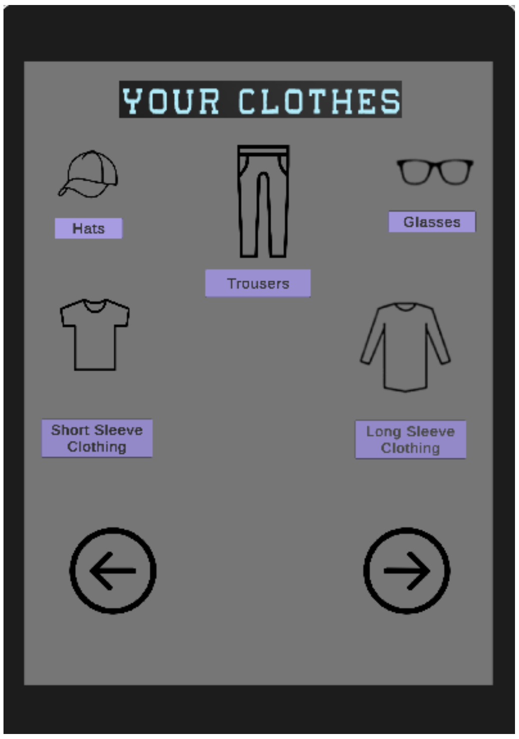


Fig 38. Screenshot showing finished UI for Clothing

If the user clicks on a genre of clothing their clothes within that genre will appear. With a button in the right-hand corner to upload more.

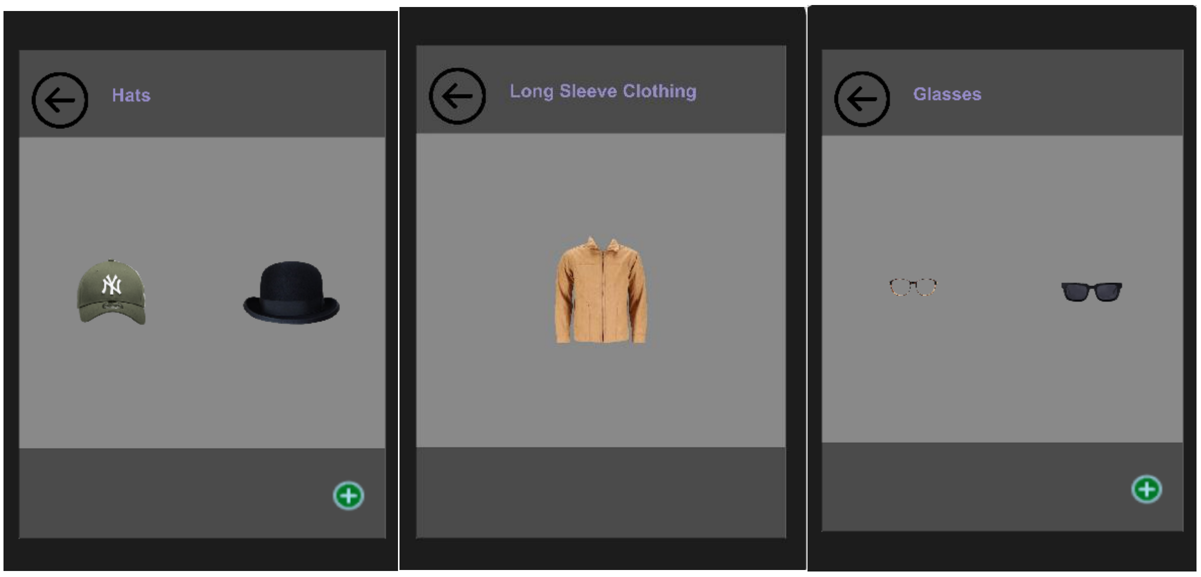


Fig 39. Screenshot showing finished UI for Clothing Library features.

Once the user clicks on the clothing to select it , a selection saved message is displayed.

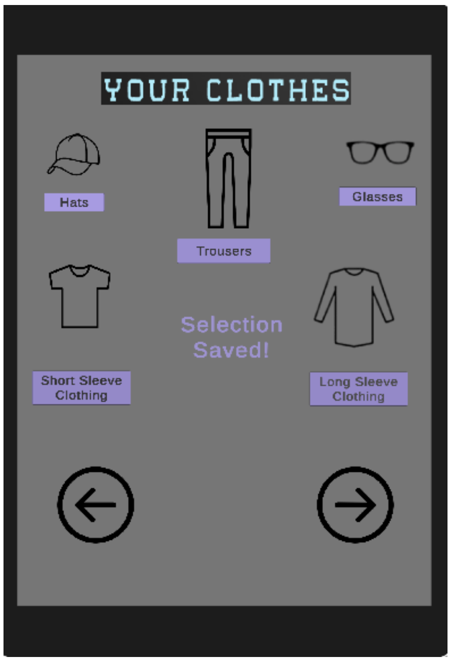
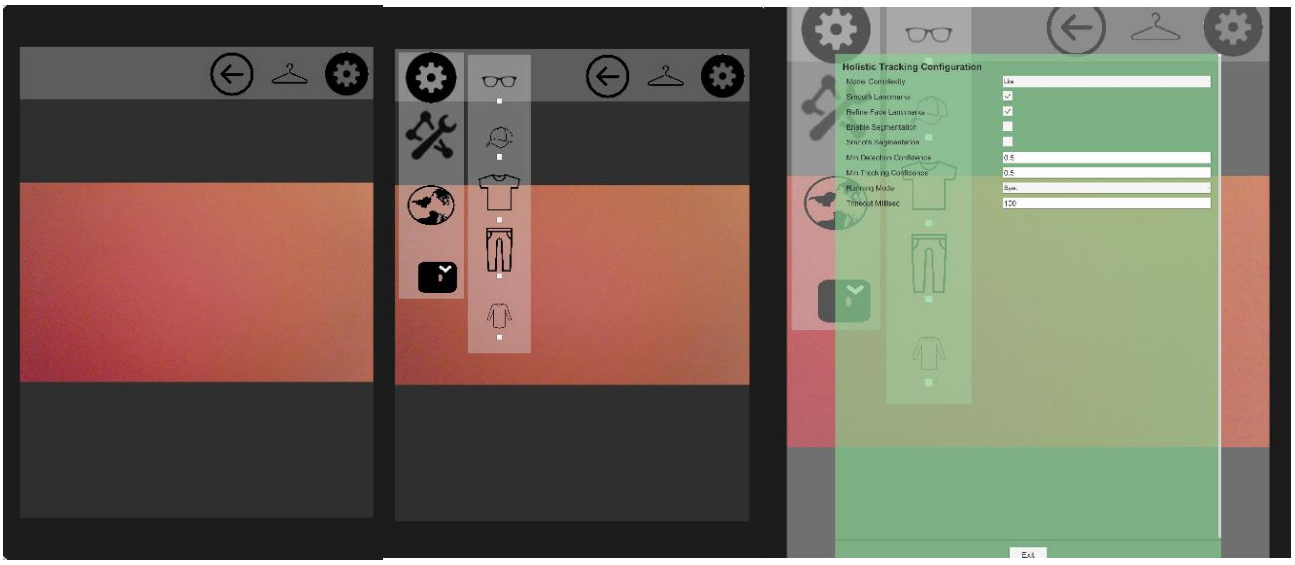


Fig 40. Screenshot showing finished UI for Clothing Library features.

After which the user would click the forward /next button to start AR. Within the AR scene there are a few customization options. A few of which include disabling clothing and changing camera settings which the user can hide/unhide with their respective icons.



Holistic Tracking Configuration

Model Complexity	Lite
Smooth Landmarks	✓
Refine Face Landmarks	✓
Enable Segmentation	
Smooth Segmentation	
Min Detection Confidence	0.5
Min Tracking Confidence	0.5
Running Mode	Sync
Timeout Millisec	100

Fig 41. Screenshot showing finished UI for Main Scene and its features.

5.2 Augmented Reality and Tracking

The tracking capabilities of the app are adequate , I noticed sometimes when lighting conditions are subpar it tends to lose tracking of the users points for a second or two. After which it restarts it. This would be expected as its only tracking the body through the camera feed without any special sensors so any disruption in the quality of the stream could affect the functionality of the app. In my experience it depends on a lot of the quality of the camera you are using. The higher the megapixels and the more detail it can capture the better the tracking is.

The Augmented Reality aspect of the app is functional but does have its drawbacks. Since I opted to use 2D images the user can only freely turn around their z and x axis for the app to look realistic. As soon as they rotate more than 15-30 degrees around the y axis and my solution start to glitch. I don't find this to be that significant as if the user turns more than that they would not be able to look at the screen anymore.

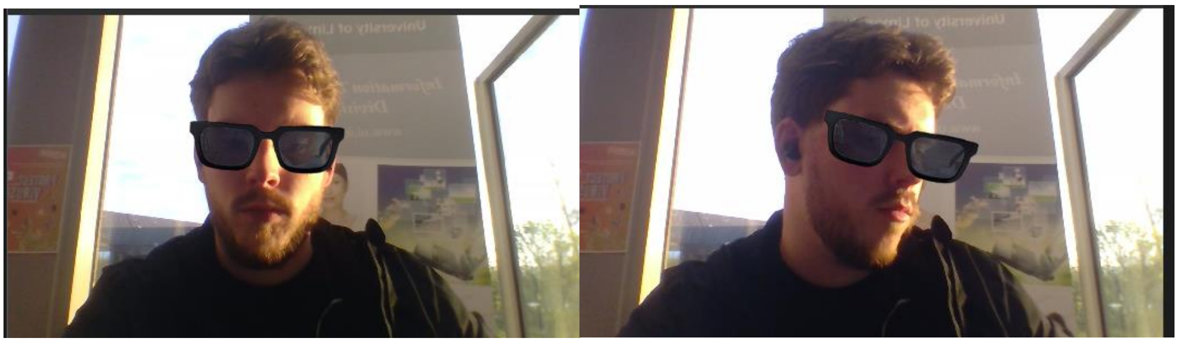


Fig 42. Screenshot depicting Limitations of Augmented Reality aspect.

To use 2D images was a decision I had to make in order to give the user a more versatile and better experience compared to 3D. This was due to the fact that for 3D a user would have had to upload a 3D model / obj. file which not many ordinary user would know how to. Therefore, my solution was to use images as a user could get an image of a product they like from a site and just download it to their device , after which they can easily upload it to the app and try on the apparel.

5.3 Technology and Methodology

In my project I've used 2 different types of tracking which are based on the framework mentioned in chapter 3 section 2 , these include Holistic tracking and Iris and Face Detection.

Holistic tracking is a computer vision solution that uses the discussed machine models to track multiple body parts, including face, upper body, and hands, in real-time video streams. It provides 3D coordinates of body landmarks such as nose, eyes, ears, shoulders, elbows, wrists, hips, knees, and ankles. It uses a combination of deep neural networks and traditional computer vision techniques to achieve high accuracy and robustness even in challenging scenarios such as occlusion, lighting changes, and camera motion. [15]

While face detection also uses the same solution it is slightly different as it provides the bounding box coordinates and key facial landmarks of each detected face, including the eyes, eyebrows, nose, mouth, and jawline.[11]

Now I've given a deeper looks into the technology behind the tracking implemented within my project , I'm going to explain the methodology used to achieve the augmented reality effect.

My main methodology was using vectors to perform the calculations. For example, I had the tracked points outputted by the framework which in this case was Holistic Tracking. After creating global variables for the tracked point within my script I was able to manipulate my target apparel based on the movement of those points. In particular, I used NumPy arrays to represent the vectors and perform vector operations such as dot product, cross product, and normalization. These operations are commonly used in vector calculus and linear algebra.

5.4 Compare and Contrast

During my market assessment I have mentioned a few apps which I found to be similar to what I set out to create. Now that I have walked you through my final product I can compare and contrast it with what's available on the market.

YourFit vs My App

My app and the 3DLook app differ in their methodology and technology. My app uses 2D images for the augmented reality aspect, which limits the users' ability to rotate their view of the clothing. In contrast, 3DLook uses a 3D model to allow for more rotation and movement of the clothing. However, my app has an advantage in terms of ease of use, as users can simply upload an image of the clothing, they want to try on rather than having to find a 3D model.

In terms of technology, my app uses Mediapipes solutions for machine learning and processing pipelines, while 3DLook uses a combination of computer vision, machine learning, and advanced 3D algorithms. Additionally, 3DLook's technology includes body scanning capabilities to ensure a more accurate fit for the clothing, while my app focuses on tracking body parts through Holistic tracking and Iris and Face Detection. Overall, YourFits app does outperform mine due to the fact that it has superior technology.

Metail (eTryOn) vs My App

In contrast with my app , the eTryOn app on is an e-commerce app that allows users to virtually try on clothes from various online retailers. The app uses a combination of computer vision, machine learning, and artificial intelligence technologies to create a virtual 3D avatar of the user and superimpose virtual clothing on it in real-time. The app also has an automatic fit algorithm that ensures that the clothes fit the user's body accurately. Compared to my app, eTryOn is more advanced in terms of technology and methodology as it uses computer vision, machine learning, and artificial intelligence technologies to create a 3D avatar of the user and superimpose virtual clothing on it. It also has an automatic fit algorithm that ensures the clothes fit the user's body accurately. My app, on the other hand, uses 2D images that could give the user a more versatile and better experience, and its tracking capabilities are adequate.

Zeekit vs My App

Here are some similarities between my app an Zeekit. Both apps use augmented reality to allow users to try on clothing virtually. Furthermore, both apps have a user interface that allows users to browse and select clothing items ,both apps use computer vision solutions to track the user's body and movements. Both apps have customization options for the AR scene.

Some of the difference between the two are:

My app uses 2D images for clothing items, while Zeekit uses 3D models. Zeekit allows users to see how clothing looks from different angles, while my app has limitations in this regard. My app uses MediaPipe solutions, while Zeekit uses its own technology. Zeekit has social sharing features and integration with e-commerce platforms, while myapp does not.

Overall, both apps have their strengths and weaknesses. Mine has a simpler interface but has limitations in terms of the AR experience. Zeekit has a more advanced AR experience and additional features such as social sharing and e-commerce integration but may be more complicated for some users to navigate.

6. Conclusion

I’ve set out to create an app that will let users try on different pieces of apparel , I’ve concluded that the project has been successful in achieving its intended goal of allowing users to try on multiple genres of clothing in an augmented reality setting. The app has a user-friendly interface that is easy to navigate and has a universal theme and colour palette throughout. Users can easily upload and select clothes from the clothes library, and the augmented reality aspect of the app is functional, allowing users to try on clothes virtually.

However, the app does have some drawbacks, particularly in the use of 2D images for clothing. This limits the realism of the experience and can make it difficult for users to gauge how a particular piece of clothing would look on them. Additionally, the tracking capabilities of the app are adequate but can be affected by lighting conditions and the quality of the camera being used.

When compared to other similar apps in the market, the app has some unique features, such as the ability to try on multiple genres of clothing and the integration of Firebase for user sign-up and login. However, the use of 2D images for clothing puts it at a disadvantage compared to other apps that use 3D models or virtual try-on technology. Overall, the augmented reality app has been successful in achieving its intended goal and provides a fun and engaging way for users to try on clothes virtually. However, there is still room for improvement, particularly in the use of more advanced technology for clothing visualization and tracking.

7. References

[1]Statista. (n.d.). *Main reasons to return clothes bought online 2021*. [online] Available at: https://www.statista.com/statistics/1300981/main-reasons-return-clothes-bought-online/.

[2]Runeberg, S. (2021). *How businesses can redesign online shopping to fight the environmental menace of free returns*. [online] Fast Company. Available at: https://www.fastcompany.com/90701492/how-businesses-can-fight-the-environmental-menace-of-free-returns.

[3]Kipper, G. (2013). *Augmented reality : an emerging technologies guide to ar*. Boston Syngress Publishing.

[4]Vossle (2021). *The difference between Marker based & Markerless Augmented Reality*. [online] Medium. Available at: https://vossle.medium.com/the-difference-between-marker-based-markerless-augmented-reality-5d294ebccb76 [Accessed 19 Apr. 2023].

[5]Carter, R. (2022). *Augmented Reality Statistics To Know In 2023*. [online] XR Today. Available at: https://www.xrtoday.com/augmented-reality/augmented-reality-statistics-to-know-in-2023/.

[6]www.youtube.com. (n.d.). *YourFit by 3DLOOK*. [online] Available at: https://www.youtube.com/watch?v=GkjPJzLbTJc&ab_channel=3DLOOK [Accessed 19 Apr. 2023].

[7]Look, 3D (2021). *Mobile Body Scanning & Measuring Technology | 3DLOOK*. [online] 3DLOOK. Available at: https://3dlook.me/technology/.

[8]Beauchamp, D. (2018). *Shopify AR Makes Shopping in Augmented Reality a Reality for Small Businesses*. [online] Shopify. Available at: https://www.shopify.com/blog/shopify-ar.

[9]Metail (n.d.). *EcoShot*. [online] Metail. Available at: https://metail.com/products/ecoshot/.

[10]www.youtube.com. (n.d.). *EcoShot Model Images - 3 minute demo by 3D fashion designer Joneien Johnson*. [online] Available at: https://www.youtube.com/watch?v=H9NFqJF7sbk&ab_channel=EcoShot [Accessed 19 Apr. 2023].

[11]Zeekit (n.d.). *Zeekit*. [online] zeekit.me. Available at: https://zeekit.me/home/solutions [Accessed 19 Apr. 2023].

[12]Mediapipe (n.d.). *MediaPipe*. [online] mediapipe.dev. Available at: https://mediapipe.dev/.

[13]Mediapipe (n.d.). *MediaPipe Solutions guide*. [online] Google Developers. Available at: https://developers.google.com/mediapipe/solutions/guide [Accessed 19 Apr. 2023].

[14]uk.mathworks.com. (n.d.). *RANSAC*. [online] Available at: https://uk.mathworks.com/discovery/ransac.html.

[15]GitHub. (2023). *google/mediapipe*. [online] Available at: https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md [Accessed 19 Apr. 2023].