Real-Time Augmented Reality for Virtual Try

A Project Report
Presented to
The Faculty of the College of
Engineering

San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Software Engineering

By
Pragya Gautam, Mojdeh Keykhanzadeh, Sithara Krishna Murthy, Vidhi Shah

May 2020

APPROVED

DocuSigned by:
Vijay Kumar Eranti
8945C38D2DB8450...

Vijay Kumar Eranti, Project Advisor

Dan Harkey, Director, MS Software Engineering

Xiao Su, Department Chair

ABSTRACT

Real-Time Augmented Reality for Virtual Try On
By Pragya Gautam, Mojdeh Keykhanzadeh, Sithara Krishna Murthy, Vidhi Shah

The way that we shop nowadays has changed by the E-commerce industry. E-commerce overtakes a major part of retail as online shopping has become a new normal. According to Statista, a global no.1 business data platform, online apparel and accessories sales in the United States are expected to reach 146 billion dollars in 2023 from 96 billion dollars in 2017. Experiencing in-store shopping in the comfort of one's home is now becoming more viable by companies which provide virtual try-on experience such as Jeeliz, Lenskart, Wannaby, Warby Parker, Sephora and MAC cosmetics.

However, current virtual try-on models only provide a specific kind of product such as eyewear, shoes or makeup. It is not convenient for customers to hop from site to site to try-on a variety of products. For instance, unlike Jeeliz that only provides eyewear try-on, Wannaby provides more options such as shoes, accessories or nail polish but no other products. Most of these models are more based on pre-captured images. Hence, there is a great need for developing an application which allows a real-time virtual try-on feature for a catalog of products, while preserving the privacy of a customer.

In this project, we are using various existing deep neural network approaches, like CLMtrackr, and VGG-19, to develop a model that can help build a virtual try-on system. The user-friendly web application will be a single spot to virtually try three different things

together i.e. hair, glasses and clothes. The system will apply video transformation on a frame-by-frame basis to enhance and secure the customer experience. Later, the web-application will be hosted over the google cloud. This will help in tracking and imprinting of objects in real-time in a privacy friendly way thus allowing the customer to be utterly satisfied with their purchase.

Acknowledgments

The authors are deeply indebted to Professor Dan Harkey and Professor Vijay Kumar Eranti for their invaluable comments and guidance in the course of this project.

Table of Contents

List of Figures

List of Tables

Chapter 1. Project Overview

Introduction

Purpose

Virtual try-on is becoming a way to make online shopping more appealing as it enhances interaction between customers and E-commerce sites. Virtual try-on is enhancing customers' experience by allowing them to try-on products even while browsing products online, which traditionally could not do so. The goal of the Real Time Augmented Reality for Virtual TryOn project is to provide E-commerce customers with a system that allows them to virtually try on a variety of products in real time while preserving their privacy. By simply accessing the site, customers can experience try-on features with their webcam live video and image. It can run on any platform by having a capability of cross-platform support.

Problem Description and Solution

The E-commerce industry has continued to become popular. There are sites that provide virtual try-on experiences for specific kinds of products such as eyewear, shoes, accessories and makeup. However, there is no one site available for customers capable of providing a variety of products to meet their needs. It is also time consuming for customers to go to the store and try specific products, and if they like then they purchase.

Besides this, even more time consuming for them, is to return the product if they are not satisfied with their purchase. To avoid such a hassle, many customers prefer to switch to online shopping. Try-on models provide them an ability to see how the product looks on

them before deciding to purchase. However, there are some challenges with current try-on models such as preserving privacy and offering a single location for catalog of products.

The Real Time Augmented Reality for Virtual TryOn solves these challenges by using deep learning models that are pre-trained for facial landmark detection, semantic segmentation and pose estimation. The application also applies 3D animations by using libraries like Three.js and WebGL. It applies video transformation on a frame-by-frame basis to secure and enhance customer's experience. Most of the features are running on the client side to protect the privacy of the customers. Also, the system does not collect data and store it on an external server from the customer for glasses tryon and hair segmentation.

Proposed Areas of Study and Academic Contribution

The E-commerce industry has grown vastly, however, there are still challenges with the current virtual try-on models. Retails provide a virtual try-on facility for only one kind of a product, such as make-up, shoes, jewelry or eyewear. In the previous chapter we have touched upon some of the research that has already been conducted in this area. Though this project we would like to enhance the previous works to give a better user experience for virtual try on applications.

As a customer, it is extremely time consuming to drive to the store, try-on different products and stand in line to purchase the few items that we like, not to mention the entire hassle of returning items as well. To avoid this situation many customers have started to switch to purchasing items online. However, this does not come without its own problems. For example, it becomes really difficult to purchase items online without ever trying them on, hence, came along the concept of virtual try-ons. Some of the major issues with the existing try-on models are that they are not preserving the privacy of customers and do not offer a single location where customers can shop for a catalog of products.

The purpose of this project is to use deep learning models for facial landmark detection, semantic segmentation and pose estimation along with 3D object animations. The system will apply video transformation on a frame-by-frame basis to secure and enhance the customer's experience. To protect the privacy and security of the customers using the application, the models will be running on the client side and hence no data will be

collected from the user and stored on external servers. This web-application will be hosted over the cloud. It will help in tracking and imprinting of objects in real-time in a privacy friendly way thus allowing the customer to be utterly satisfied with their purchase. This way the customer does not have to go through the hassle of spending time on driving to the store to first try-on the items they would like to purchase or return items that do not appeal to them.

Current State of the Art

The most widely well-known technique for image generation as well as expression detection is GANs. GANs consist of two neural network models, a generator and a discriminator which are trained using adversarial learning ideas. The generator model generates new samples while discriminator discriminates real data samples from generated samples. The IEEE paper [1] describes image-to-image translation as an application of GANs. The goal of it is to map input image to output image. Two-Pathway GAN (TP-GAN) is used for face recognition. TP-GAN requires paired training data of frontal view image and different posing image. In the absence of paired data, CycleGAN is used for a variety of image-to-image translation activities such as subject transfiguration and style transfer.

Although GANs are good generative modeling, they are challenging to train. In the book [2], David Foster illustrates challenges of GANs and solutions to them. The challenges are oscillating loss, mode collapse and uninformative loss. The rise of the Wasserstein GAN (WGAN) enhanced stability of GANs by diminishing likelihood of mode collapse. WGAN removes the difficulty of finding balance between training of generator and discriminator by training the discriminator several times between generator updates. Transfer learning also is a technique that allows to transfer knowledge to new tasks from already learned source tasks. According to the article [3], transfer learning is an improvement to traditional algorithms which use isolated tasks. Transfer learning makes machine learning efficient,

just like human learning. Video transform is an area of transfer learning which allows transformation of videos on a frame-by-frame basis.

Chapter 2. Project Architecture

Introduction

This project uses deep neural network application programming interfaces for enabling virtual try-on. Since currently available try-on applications in the market are only incorporating one try-on feature at a time, the architecture of our project mainly revolves around including all three try-on features (glasses, clothes, hair) in a single robust application. Overall architecture diagram is shown below and each of the sub-components are described in detail in the following section.

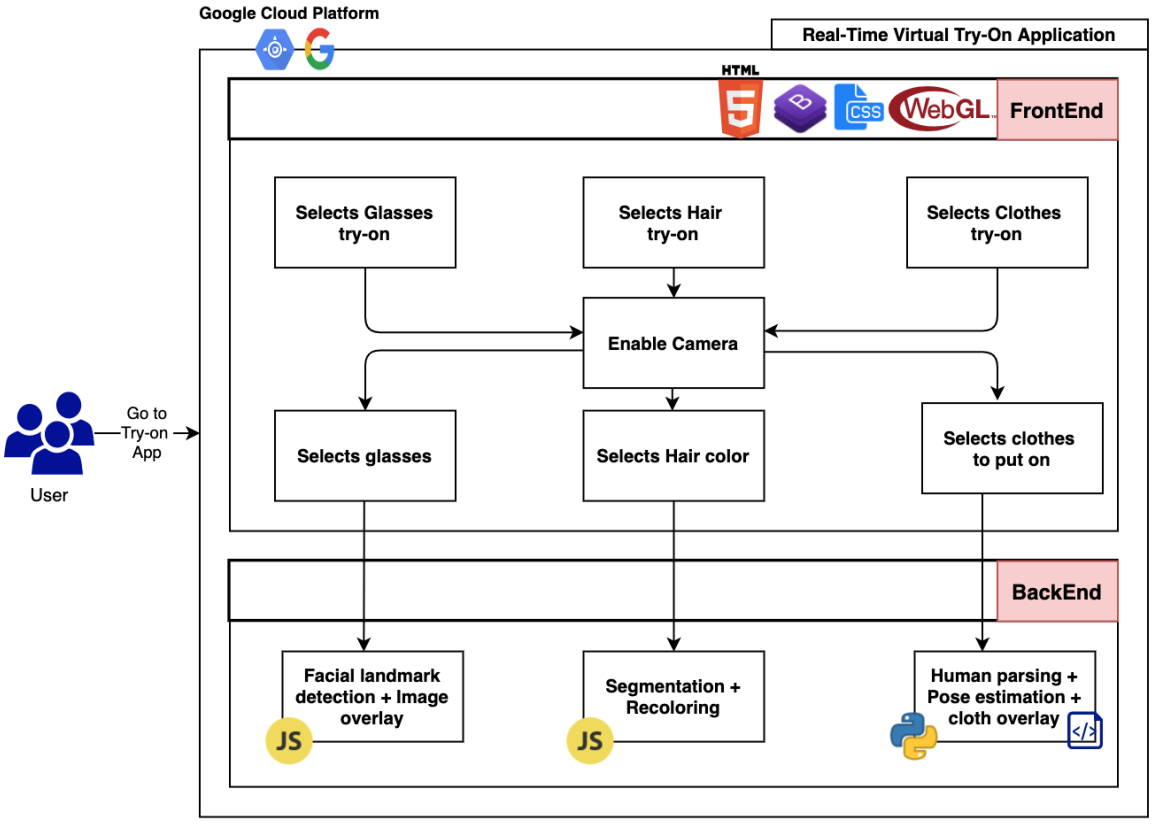

Figure 1 Architecture Diagram

Architecture Subsystems

Google Cloud Platform (GCP): We will be hosting our try on application under google cloud platform by creating a compute engine server. It will be a public cloud-based machine whose services will be used to leverage its resources to empower the application in order to reach more customers. GCP offers a virtual machine hosting service like AWS, but its main service is development and deployment of containerized applications. In this project, GCP will primarily be used for computational purposes.

Front-End: The interactive web application interface is built using HTML5, CSS, and Bootstrap. WebGL JavaScript API is also used for rendering 2D and 3D interactive graphics. First, the user goes to a try-on application and selects the tab for the features he/she wants to use out of glasses, hair and clothes. The application will then ask the user to allow the camera access. Real-time video is then rendered to the user. Next, the user will select either a pair of glasses/hair color/cloth to perform the try-on operation. The request is then sent to the backend API based on the selection made.

Back-End: The API request received from the front-end is processed by back-end. For glasses try-on, it will call face detection and image overlaying APIs built using JavaScript and send back the output on the UI by putting the frames on the user's face. In case of virtual hair coloring, the API request is sent to perform hair segmentation and hair recoloring for the selected hair color. These utilities are built in JavaScript and use WASM too. The output is rendered to the user by putting the colored hair. Clothes try-on API is

built using python programming language. A single shell script runs that first takes an input image and sends it to the human parsing model, then to pose estimation to detect body points. In the end, clothes/image overlaying is performed generating the final output image with clothes selected by the user.

Chapter 3. Technology Descriptions

Client Technologies

JavaScript & jQuery

JavaScript, is a high level programming language that is mostly just-in-time compiled. It uses curly-bracket syntax, first-class functions and is object-orientation. Similar to HTML and CSS, JavaScript is one of the most popular technologies used for the internet. JavaScripts are usually embedded in HTML files and interact with the Document Object Model.

JavaScript allows interactive web pages and is a major part of web applications. Most websites use JavaScript libraries or web application frameworks for their client side pages. JS is mostly used for the client side pages, where 95% of the web browsers have a dedicated JS engine to execute the JavaScript code. One of the most popular libraries that is used is the jQuery library, it is used by over 70% of the websites.

jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. This makes it much easier for users to navigate a document, handle events and create animations. The most used functionality is its ability for developers to create plug-ins on top of JavaScript. This modular approach of jQuery allows users to create powerful dynamic web pages and applications.

HTML, CSS & Bootstrap

HTML is one of the building blocks of the Web as it defines the structure of web content. Other technologies besides HTML, such as CSS and JavaScript are generally used to describe a web page's appearance and functionality, respectively.

To describe a document such as HTML, originally written in the form of a markup language, one can use a style language called Cascading Style Sheets. One of the most widely used CSS frameworks is Bootstrap.

Bootstrap is a free, open-source framework used for responsive front-end development. It contains CSS design but can also support JavaScript based design templates. Its main purpose is for styling the front-end webpage with the use of elements such as navigation bars, buttons color, size, font, and layout. Bootstrap also has jQuery plugins which are composed of several JavaScript components. These plug-ins provide user interface elements such as boxes and carousels.

Middle-Tier Technologies

Node.Js

Node.js is an open source environment that can run on various platforms like Windows, Linux, Unix, Mac OS. It mainly uses JavaScript on the server side and allows the creation of web servers using the same. Node.js also uses asynchronous programming. Commands in Node.js executes concurrently as well as in parallel. The functions within it are non-blocking and hence, it is easy to use language especially when one wants to create web servers.

Node.js follows an event-driven programming approach and allows the developers to make use of callbacks to signal success or failure in the program. When a new project is created using Node.js, developers can run 'npm i' to install all the libraries for the project and there are thousands of open-source libraries presently available for Node.js. Node.js also supports various web frameworks like Express.js, Connect, Socket.IO and many others.

Python

Python is a dynamically typed and interpreted language. It supports object-oriented, structured as well as functional programming paradigms and its features. It is also a high-level, general-purpose language and is garbage-collected.

Unix Shell Scripting

A shell is basically a command-line interpreter while shell script is a computer program to be run by a shell-like Unix or Linux. General operations performed by shell scripts are file

manipulation, printing text and doing a program execution. The file in which shell scripts are written requires a set of instructions to tell the shell what to do and when to do. The commands are executed in the sequential order.

Deployment Technologies

Google Compute Engine

We are using Google Compute Engine as our deployment tool. Google Compute engine creates virtual machines which are versatile. It can be easily configured for any number of RAMs, processors and storage space depending upon the need and usage. One gets billed only as per the usage and as university students we also get $300 promotional credits to use upfront with it. Using Google's infrastructure, Compute Engine offers great scale and performance. It has been designed to be really fast and provide consistent performance. It almost works as a local workstation where the same setup steps need to be followed to run a web application, except it lives in the cloud.

Chapter 4. Project Design

Client Design

USE CASE DIAGRAM

Glasses Try on: The customer has the possibility to access the web application, enable the video access, try on the glasses, and select from a set of different options of glasses.

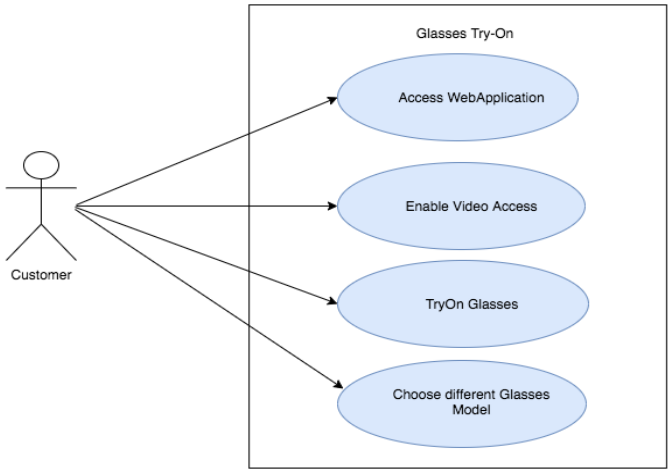

Figure 2 Glasses Try on Use Case Diagram

Change Hair Color: The customer has the possibility to access the web application, enable the video access, and try on different hair colors.

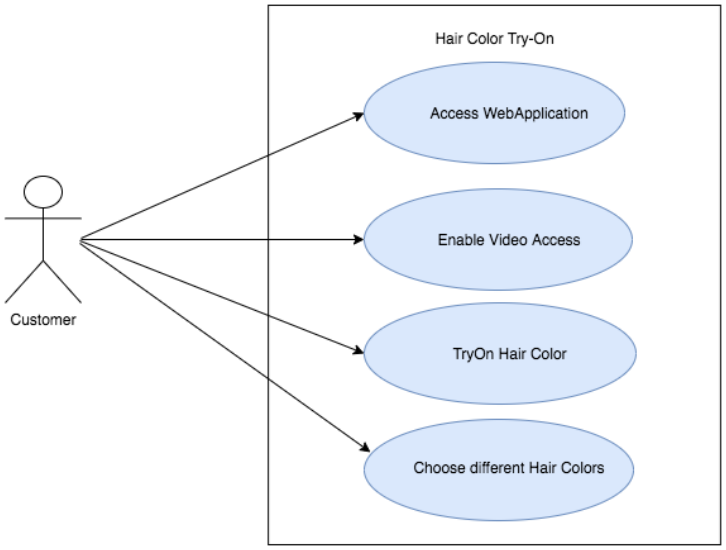

Figure 3 Hair Color Use Case Diagram

Clothes Try on: The customer has the possibility to access the web application, enable the video access, take a picture of themselves to upload for the clothes try-on and select different clothes to try on.

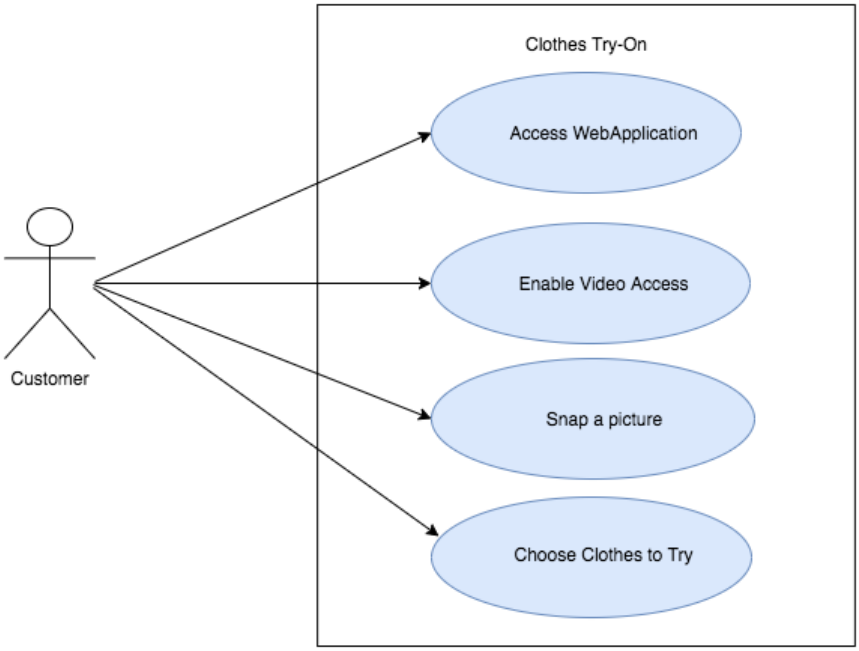

Figure 4 Clothes Try on Use Case Diagram

User Interface Mock-Ups

Home Page:

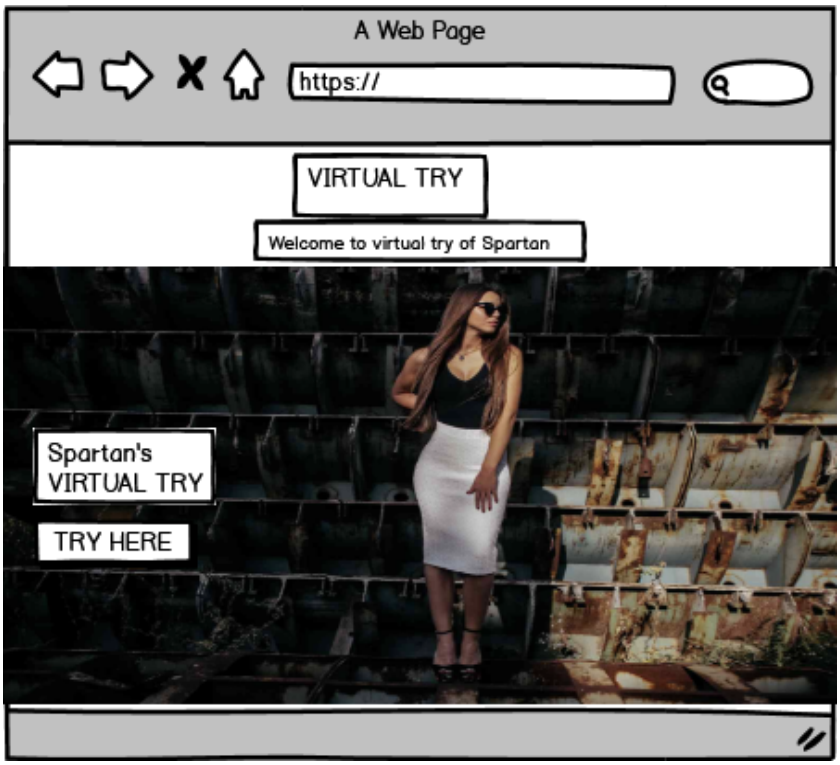

Figure 5 Home Page

Virtual Try on of Glasses:

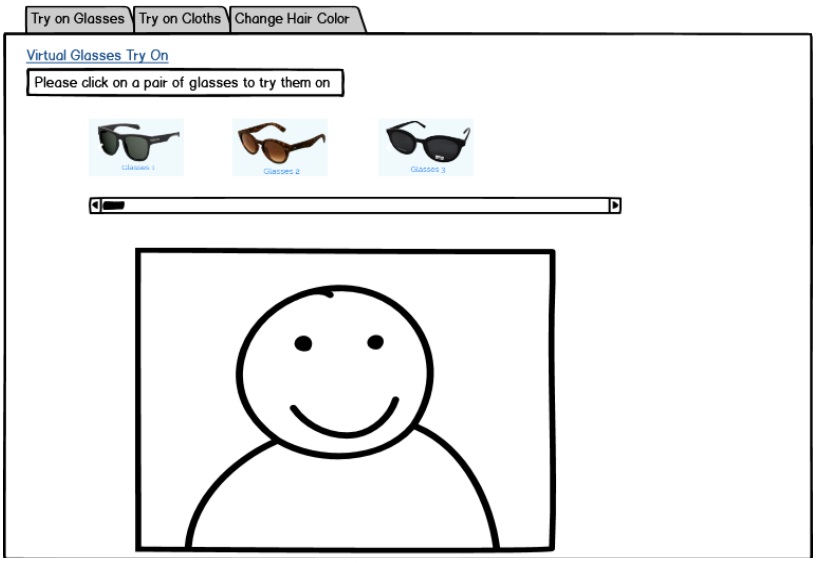

Figure 6 Glasses Try on

Virtually changing Hair Color:

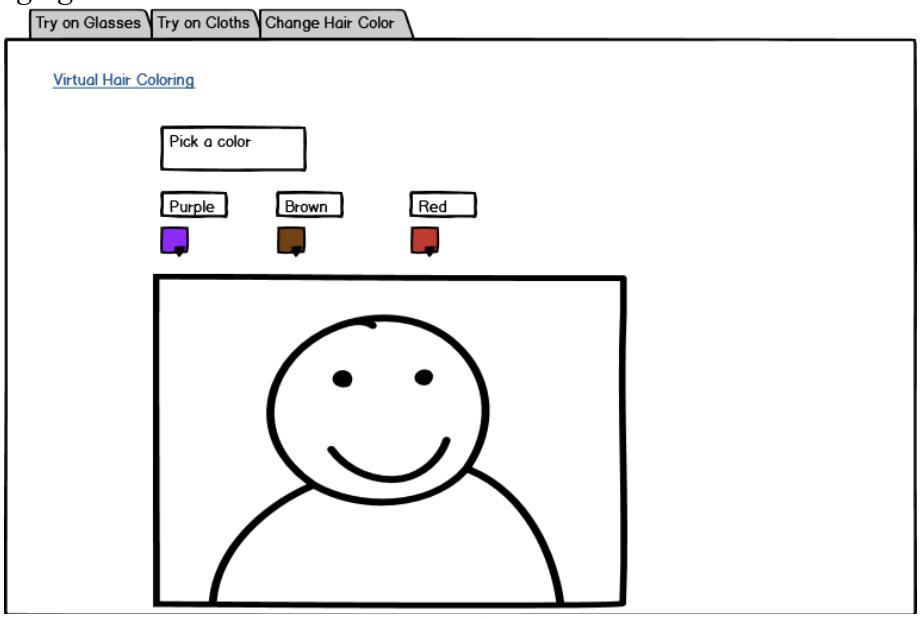

Figure 7 Hair Color

Virtual Try on of Clothes:

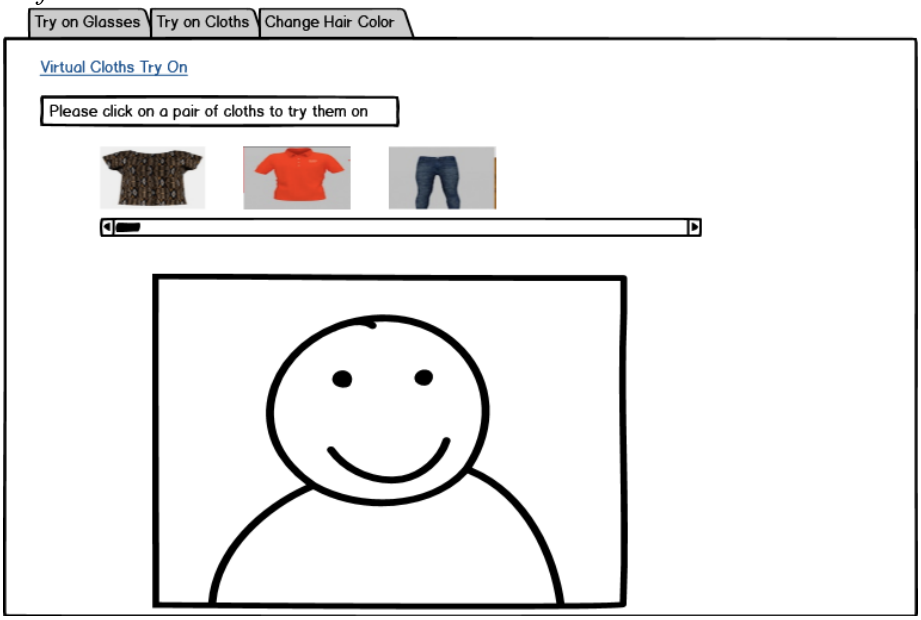

Figure 8 Clothes Try on

Flow Diagrams

As a user of the application, customers must load the application on their browsers and enable their camera to experience the real-time virtual try-on. The customer process can be depicted by the following process flow diagrams:

Glasses Try on: Once the user has granted access to the camera, the application will start with the Facial Landmark detection using the CLMtrackr, followed by overlaying of the glasses on the person. If the user does not like the glasses, they have an option to browse multiple glasses and select the ones they would like to try on.

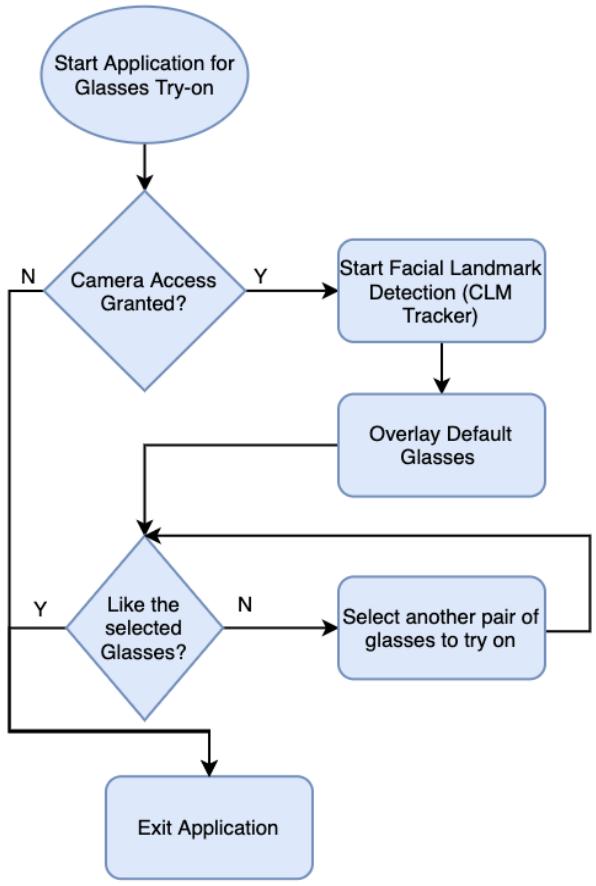

Figure 9 Customer Flow Block Diagram for Glasses Try on

Hair Color: Once the user has granted access to the camera, the application will start with the Hair Segmentation using a MediaPipe pipeline with a TFLite model, followed by changing of the hair color. If the user does not like the color, they have an option to scroll the RGB values and select the color of their choice to change their Hair Color into.

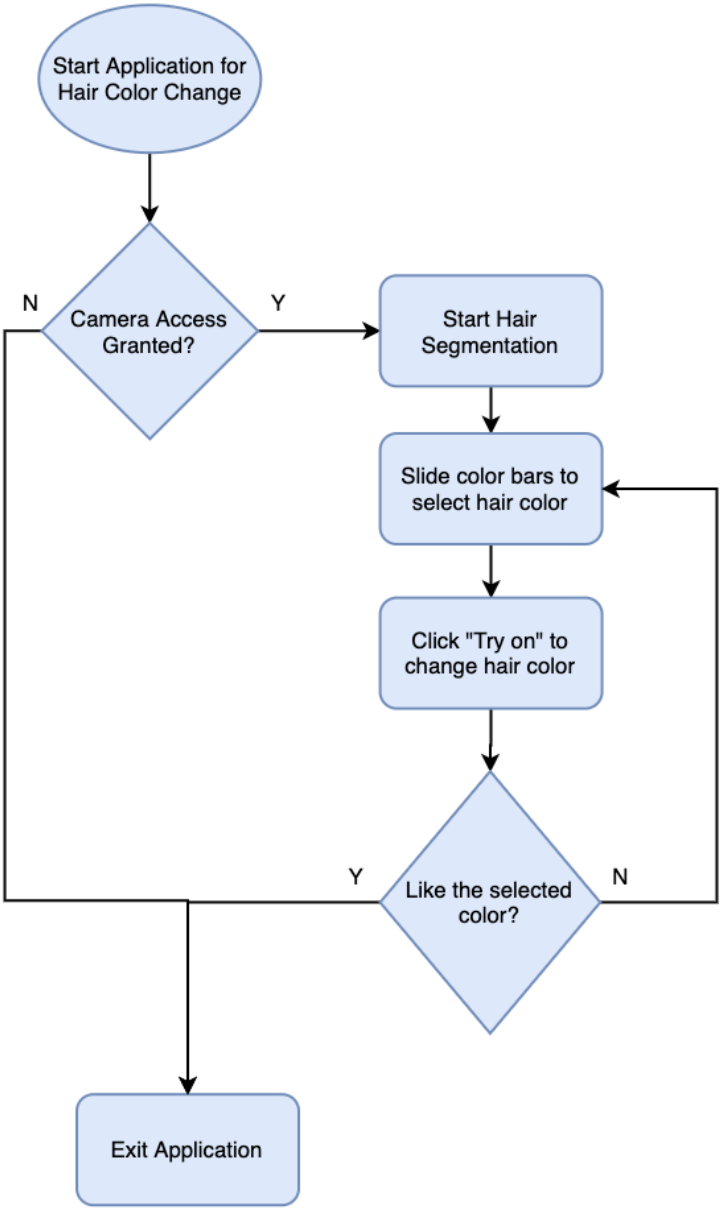

Figure 10 Customer Flow Block Diagram for Hair Color Change

Clothes Try on: Once the user has granted access to the camera, the application will start with the user being able to take a picture of themselves. This picture will then be processed, and the clothing selected by the user will be overlaid on the body. If the user does not like the clothing, they have an option to scroll and select the clothing of their choice.

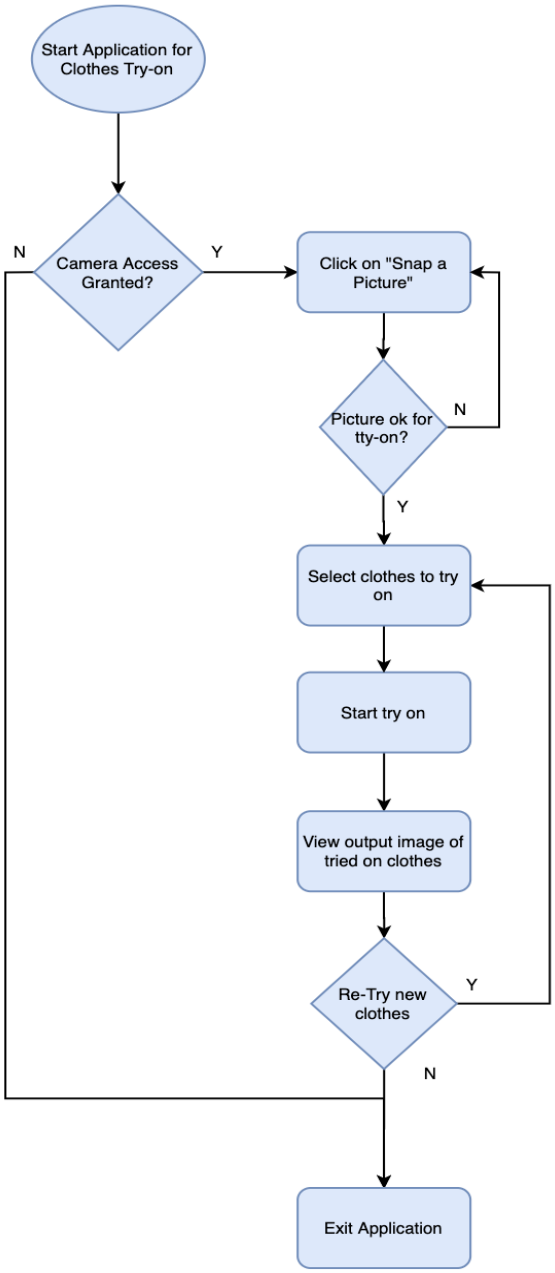

Figure 11 Customer Flow Block Diagram for Clothes Try on

Middle-Tier Design

Sequence Diagram

Glasses Try on:

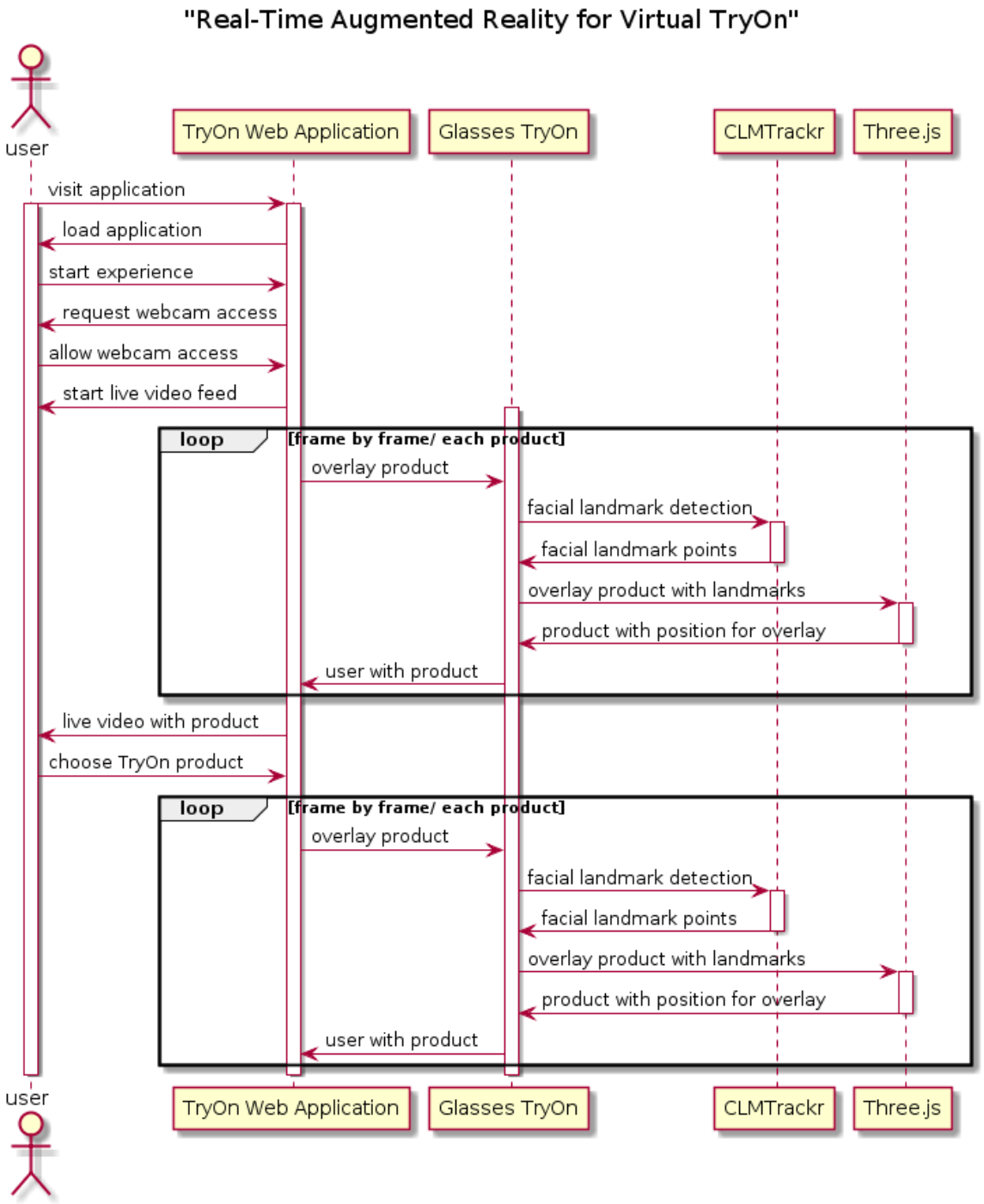

Figure 12 Sequence Diagram for Glasses TryOn

Hair Color

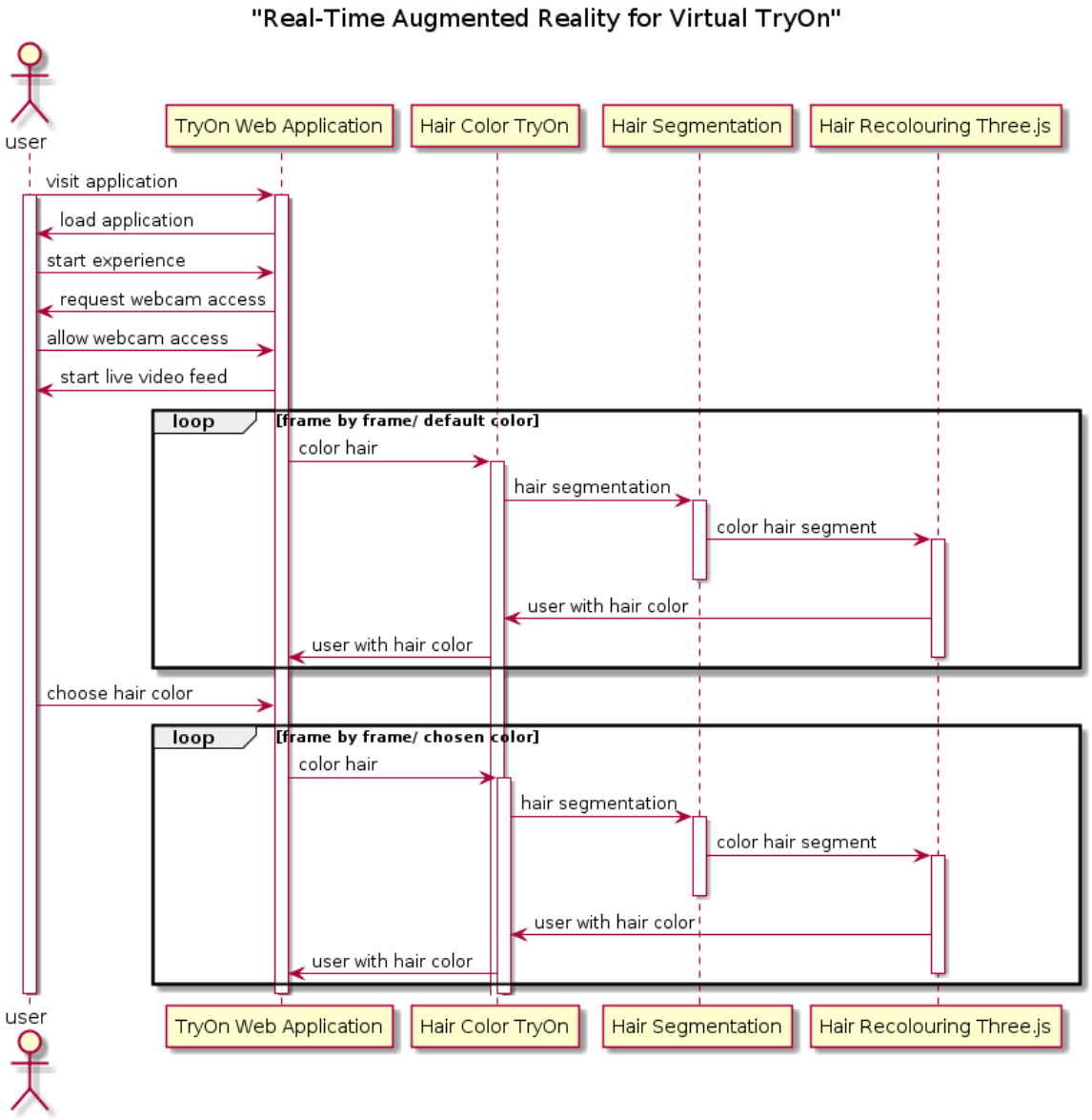

Figure 13 Sequence Diagram for Hair Color

Clothes Try on:

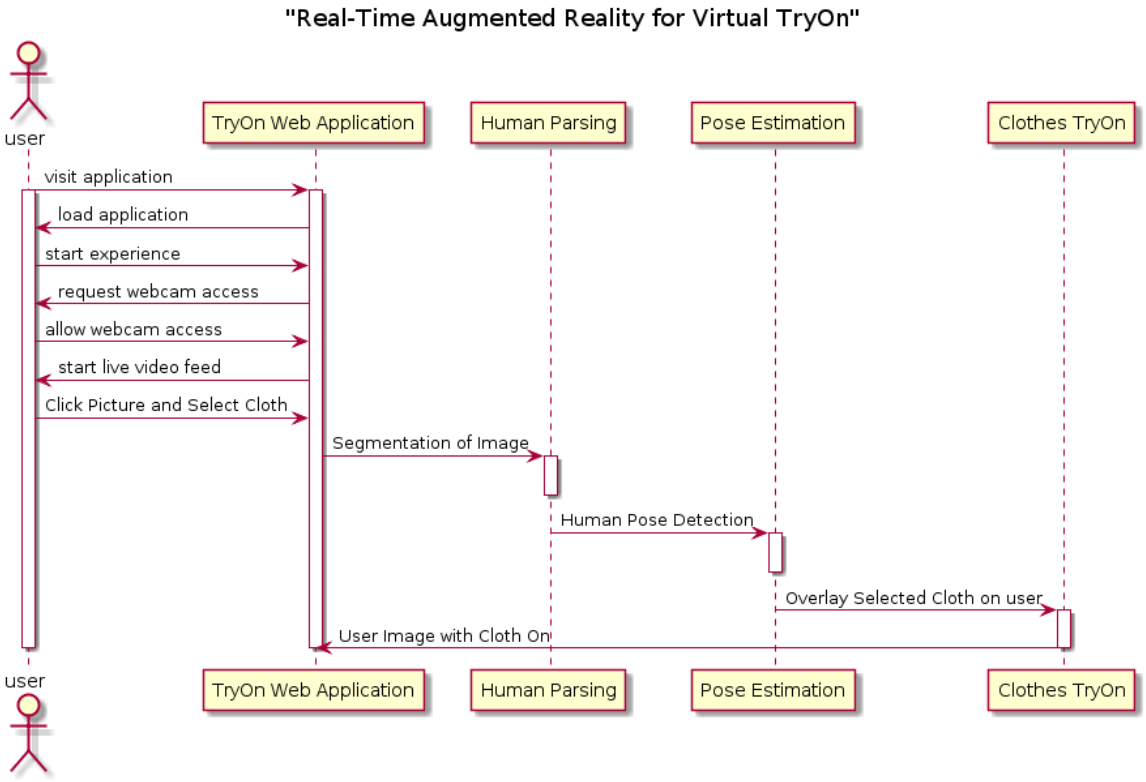

Figure 14 Sequence Diagram for Clothes TryOn

Deployment Design

For our deployment, we decided to use Google Cloud Platform (GCP). We have a Node.js web application with embedded machine learning models as major parts. The reference to our neural net models is inside our node.js app so it gives us flexibility to simply deploy node.js app on google compute engine. Figure 12 shows the deployment architecture of our node.js app.

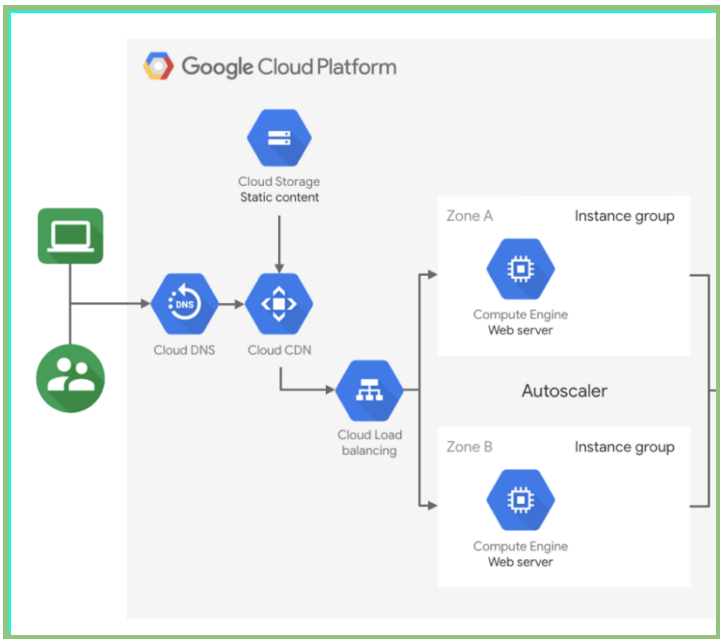

Figure 15 Deployment Diagram

Cloud DNS: Google infrastructure uses highly available DNS provider, Google cloud DNS to handle the requests for the website in the browser

Cloud CDN: Cloud CDN is a network of edge locations. Hence, for all the incoming requests, they are automatically routed to the nearest location and content is sometimes cached to deliver the best performance.

Cloud Storage: All the images, stylesheets, CSS files and other static content within the application is stored under Google cloud storage.

Cloud Load Balancing: To distribute the incoming traffic from the web application, cloud load balancing is used. It evenly divides it between multiple compute engine instances. HTTP requests are automatically handled by load balancers while HTTPS requests require SSL certificates.

Compute Engine Web Server: The entire web application, including both front-end and back-end is deployed within the google compute engine (GCE) instances. It automatically does scale the application. Instances can be started or stopped as per the need and can be grouped as a managed instance group.

Chapter 5. Project Implementation

Client Implementation

WebGL

WebGL is a JavaScript API used for creating 2D and 3D graphics in a browser without any plug-ins. As WebGL is integrated with the standards of the web it allows GPU to use image processing as part of the web page. WebGL elements can be combined with HTML elements as part of the web page or background. A browser with WebGL support is highly recommended while tracking the video. However, the library will work with any modern browser such as Chrome, Edge, Firefox and Safari.

In this project we are using WebGL for rendering of the glasses. These glasses are being overlaid as a 3D object on top of the camera real-time video canvas.

```
/* face tracker */
this.tracker = new clm.tracker({
    useWebGL: true
});
```

Middle-Tier Implementation

Facial Landmark Detection

CLMtrackr: CLMtrackr is a JavaScript library used for tracking the face and fitting facial models in videos and images. The CLMtrackr tracks and outputs the coordinate face model as an array based on the position:

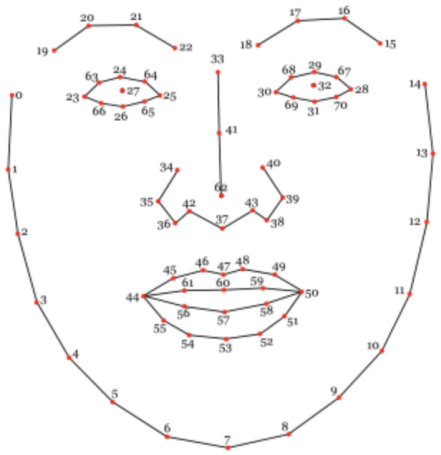

Figure 16 Coordinates

The algorithm uses 70 small classifiers to fit the face, each classifier represents a position on the face.

```
this.loop = function () {
    requestAnimFrame(ref.loop);

    var positions = ref.tracker.getCurrentPosition();

    if (positions) {
        //current distance
        var distance = Math.abs(90 / ((positions[0][0].toFixed(0) - positions[
                14][0].toFixed(0)) /
            2));
        //horizontal angle // горизонтальный угол (поворот лица)
        var hAngle = 90 - (positions[14][0].toFixed(0) - positions[33][0]
            .toFixed(0)) * distance;
        //center point
        var center = {
            x: positions[33][0],
            y: (positions[33][1] + positions[41][1]) / 2
        };
        center = ref.correct(center.x, center.y);

        var zAngle = (positions[33][0] - positions[7][0]) * -1;

        //allowable distance
        if (distance < 1.5 && distance > 0.5) {
            ref.changeStatus('STATUS_FOUND');

            //set positions
            ref.position.x = center.x - (hAngle / 2);
            ref.position.y = center.y;
            ref.rotation.y = hAngle / 100 / 2;
            ref.rotation.z = zAngle / 100 / 1.5;
            //size
            ref.size.x = ((positions[14][0] - positions[0][0]) / 2) + 0.05 * (
                positions[14][0] -
                positions[0][0]);
            ref.size.y = (ref.size.x / ref.images['front'].width) * ref.images[
                'front'].height;
            ref.size.z = ref.size.x * 3;
            ref.position.z = (ref.size.z / 2) * -1;
```

In our project, we make use of the CLMtrackr to identify the position of the points of the eyes so that we can overlay the 3D glasses on the face of the customer. Given the initial approximate position, the model searches each small region around each of the points and tries to find a better fit. Once a better fit is found, the model moves incrementally in the direction of the better fit, this continues until the best fit is found and the glasses are

overlaid. As it is very difficult to annotate and generate new faces for the training of the model, we have used the existing annotations from the MUCT database.

THREE.JS: Three.js is a JavaScript library and API which is used for creating and displaying 3D graphics. It provides cross-browser support. It internally uses WebGL for 3D rendering. It can be used in any application as part of the client using JavaScript. It helps with creating GPU accelerated 3D animations on the browser. It is a high-level library which helps in achieving difficult 3D computer animations on the browser with very less effort. Some of the features included in this library are scenes, effects, cameras, audio, lights, controllers, shadows, objects, shader materials and many others. Each of these can be used to create animations easily.

```
var canvas = document.getElementById("overlay");
var renderer = new THREE.WebGLRenderer({
    canvas: canvas,
    antialias: true,
    alpha: true
});
renderer.setClearColor(0xffffff, 0);
renderer.setSize(this.width, this.height);

//add scene
var scene = new THREE.Scene;

//define sides
var outside = {
    0: 'left',
    1: 'right',
    4: 'front'
};

this.images = [];
var materials = [];
for (i = 0; i < 6; i++) {
    if (this.object.outside[outside[i]] !== undefined) {
        var image = new Image();
        image.src = this.object.outside[outside[i]];
        this.images[outside[i]] = image;
        if (i === 0 || i === 1) {
            materials.push(new THREE.MeshPhongMaterial({
                map: THREE.ImageUtils.loadTexture(this.object.outside[
                    outside[i]]),
                transparent: true,
                opacity: 0.43
            }));
        } else {
            materials.push(new THREE.MeshLambertMaterial({
                map: THREE.ImageUtils.loadTexture(this.object.outside[
                    outside[i]]),
                transparent: true,
            }));
        }
```

We have implemented the overlaying of the glasses using Three.js library. To achieve overlaying of glasses on user video, we are initially using CLMtrackr for identifying the facial landmarks. This is used to identify the positions to place the glasses. A Scene is used as a setup for the glasses. Different materials are added to the scene, like MeshPhongMaterial, MeshLambertMaterial, PointLight, and CubeGeometry. Each of these materials are used to add animations to the image. Materials help in settings for shiny and non-shiny surfaces, point light to shoot light in all directions, perspective camera for managing the appearance of the image at different angles and cube geometry for identifying the position of the object. Once the scene is created, it is placed on the canvas which has real-time user video.

Hair Coloring

MediaPipe is a framework for creating machine learning pipelines. It is based on graphs. The pipelines are represented using graphs and these are used to process the input and obtain the desired output. MediaPipe runs on mobile devices, servers and workstations providing cross-platform support. It also supports mobile GPU acceleration. Audio and video streams can be used as input to the graphs. The main application of MediaPipe is, it can be used for creating machine learning pipelines with models and reusable components.

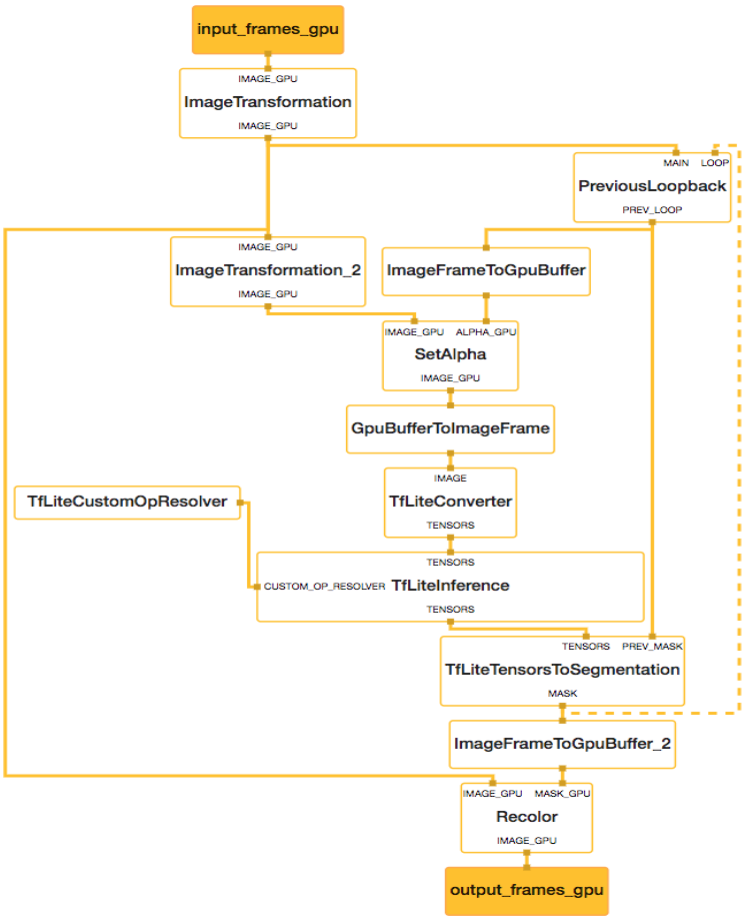

Figure 17 MediaPipe graph for Real-Time Hair Coloring

The above graph shows the flow of data through the mediapipe pipeline. It performs hair segmentation using TensorFlow Lite. It processes the video frame by frame and provides the output video with the desired hair color. The graph takes the video frame along with the selected hair color as the input. It transforms the input into a 512x512 image. It also caches the mask for each iteration and uses it along with the timestamp of the current image for the prediction in the next iteration. For the first iteration, an empty mask is sent. With RGB input and mask, the image is converted into a tensor for the Tensorflow Lite model to identify the segments of hair. The model outputs tensors that represent hair with the same width and height as of the input image tensor. Then the output tensor is decoded along with the previous mask which was cached, also the new mask is cached for the next iteration. This is used along with the input image to identify the hair segment and the desired color is applied on the image.

Hair Segmentation: Segmentation is the process of dividing an image into its segments. It helps with identifying the contents of an image. Segmentation can be used to identify objects, people, body parts, etc in an image. Similarly, a segmentation model is used to identify hair of a person in a given image. This is achieved by using a machine learning model for semantic segmentation using convolutional neural networks.

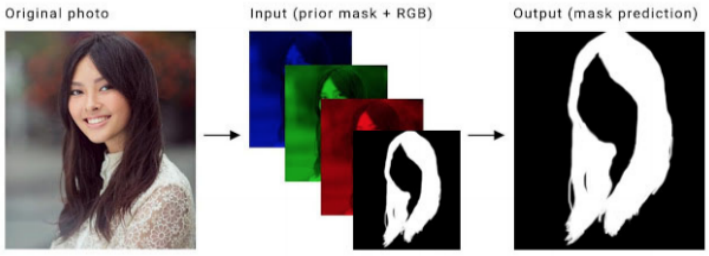

Figure 18 channels (RGB + previous frame mask) → current frame mask

The model computes a soft binary mask which separates hair from the background for each frame of the input user video. It also uses the mask from the previous frame as a fourth channel along with the current frames RGB channels, to compute the segmentation mask for the current frame. Using this information, the neural network predicts the mask for the frame.

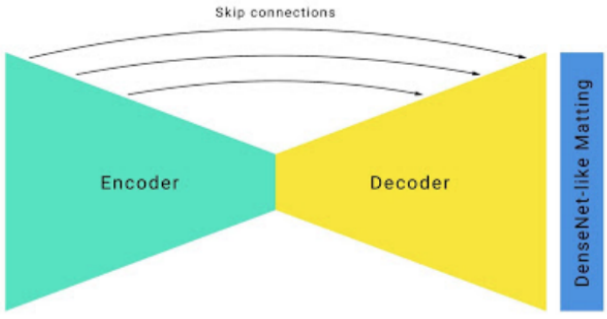

Figure 19 Network Architecture

The model is built on a standard hourglass segmentation network architecture with skip connections. It is customized with changes for better improvements. Some of them are, use of big convolution kernels on the input image channels, aggressive downsampling with skip connections and large strides, and use of densenet layers.

Hair Recoloring: Hair recoloring can be done, once the segments are obtained for an image. This needs to be done by carefully selecting effects that are tuned for a specific kind of hair.

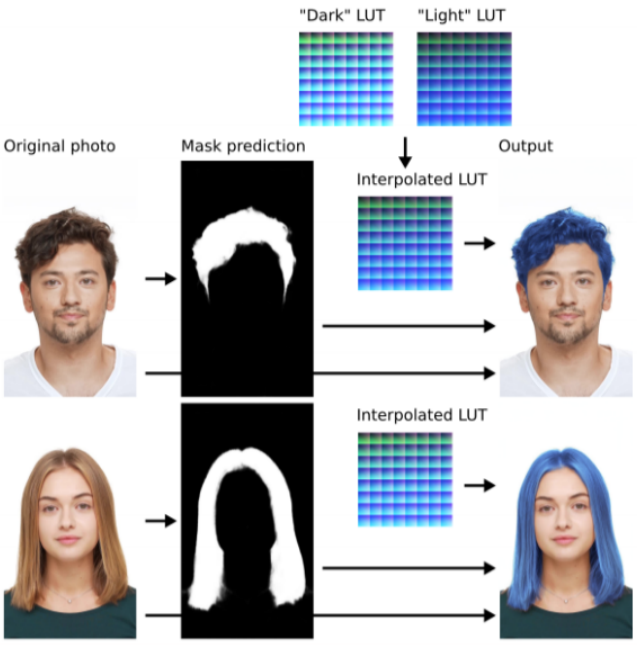

Figure 20 Hair coloring procedure

This is done in two steps, known as preparation and application. The preparation step involves selecting two images with light and dark hair colors. The intensity is calculated for both of them and they are tuned to project the desired effect. These adjustments are tabulated and stored.

The application section takes the segmentation from the model output. Uses the image and mask details and calculates the hair intensity by using the lookup table and generates the image with desired hair color.

Clothes Tryon

In our application, clothes try-on is a virtual room fitting service for e-commerce. It allows users to virtually try-on the clothes by allowing access to their camera. Users can turn-on the camera and snap a picture which will be sent first to the human parsing model, then to the pose estimation model and in the end it will flow through the clothing transfer model. Each of these stages have their own implementation techniques and deep learning models used within it. They are described below in detail:

Human Parsing: This model uses JPPNet for human parsing. JPPNet is a deep learning technique which is built on top of Tensorflow . By imposing human pose structures into parsing results as part of self-supervised structure-sensitive learning approach (SS-JPPNet) , the joint pose estimation and human parsing network can powerfully predict human parsing and pose estimation at the same time and simply solve human parsing. Self-supervised structure-sensitive learning is one of human parsing methods which is built on top of Caffe. With no need for extra supervision of labeling human joints in training , this framework can simply be injected to any neural net and improve human parsing results. We are using pre-trained models of JPPNet on LIP dataset for human parsing. The following is the input to the human parsing model and the resulting output

Figure 21 Human Parsing Input and Output

Pose Estimation: Pose-estimation is a process of determining the body points in a pose of an object or human, given an image or real-time scan. In this project, we are using a pre-trained model to perform the pose-estimation. The pre-trained model can learn the body parts using 2 methods. First one is confidence maps and second is using part affinity fields prediction. It uses the method of jointly learning parts detection and associations.

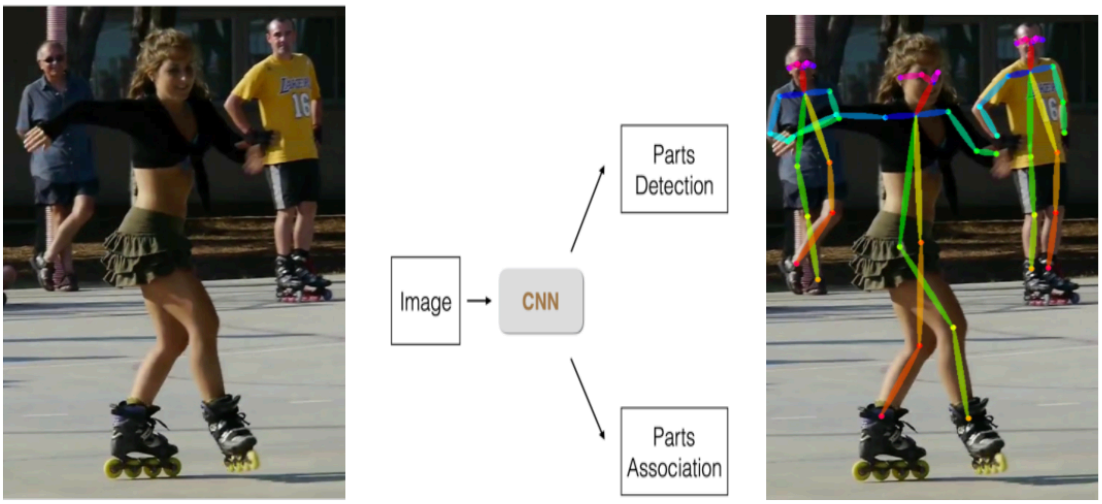

Figure 22 Input/Output images for pose-estimation

The model training for the first 10 layers was done using VGG-19 model architecture and rest with CNN. The original pre-trained caffe model was then converted to the keras model and an input image was passed to it to generate an output having pose points. Following figure shows all the body parts and color coding used:

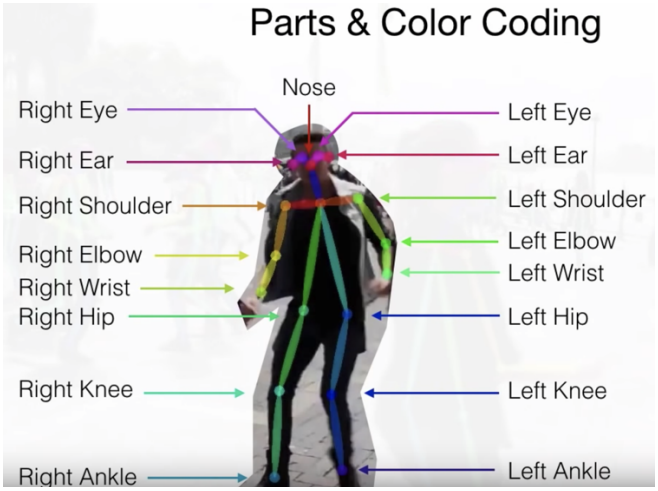

Figure 23 Parts & Color Coding pose-estimation

Clothing Transfer: As the last step of the try-on, clothing transfer is done. This will be used to transfer desired clothing on a person with coarse to fine technique. The pre-trained model from VITON, an image-based virtual try-on network is used for the same. It goes through 2 stages - stage 1 is responsible to do the inference by determining the mask to be put on a person's image based on body points and sample clothing to put on. Below is the sample output of stage 1:

Figure 24 Clothing transfer: stage-1 output

The stage-2 will do further refinement on the generated results from the stage-1 and generate the final results as shown below. Here, the second last image is the final image with cloth put-on. This final image is passed to the front-end web app to show it to the user.

Figure 25 Clothing transfer: stage-2 output

Deployment Implementation

The steps and configuration that were followed to deploy the application on google compute engine are as follows.

Create Google Cloud Project: We started by creating a new project for doing our deployment under google cloud console. The name of our project is 'try-on-app':

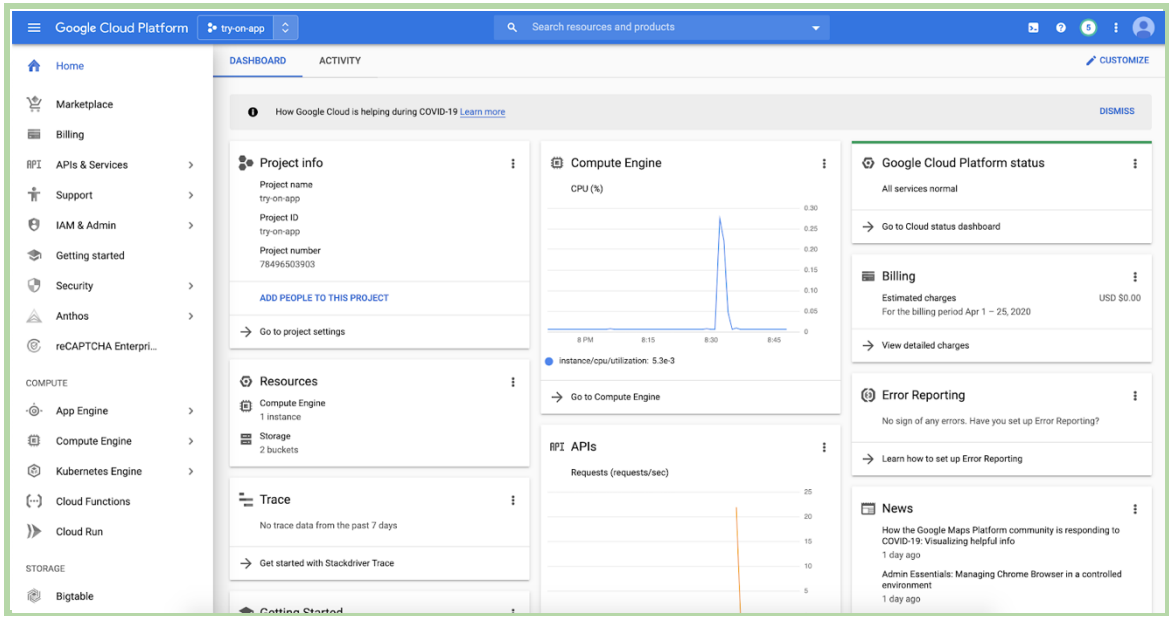

Figure 26 Create GCP Project

Create Google Compute Engine Instance: Then we started by creating a cloud compute engine instance. The Compute offers 1-96 processors, 0.6 - 624 GB, powerful disk limits and preconfigured images. We choose to have n1-standard-8 having 8 virtual CPUs and 30GB memory along with Ubuntu 18.04 as the operating system. We also kept the firewall open for both HTTP/HTTPS traffic. The following figure shows the exact configuration done:

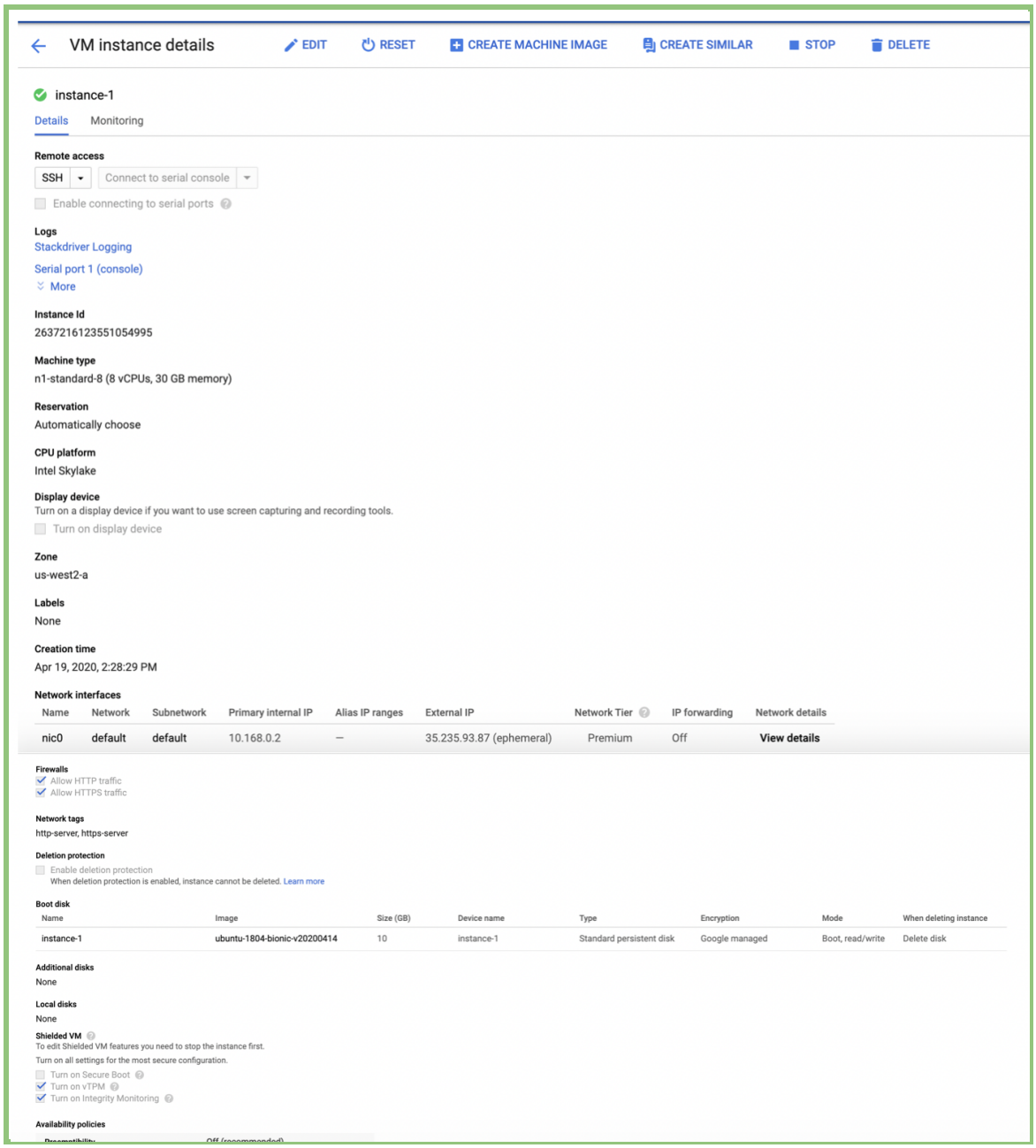

Figure 27 Create GCE instance

SSH into GCE instance: The created instance can be viewed under the cloud console itself.

To start setting up the application in the cloud, connect to the instance using the SSH link.

It will open up a browser window as shown in below screenshots:

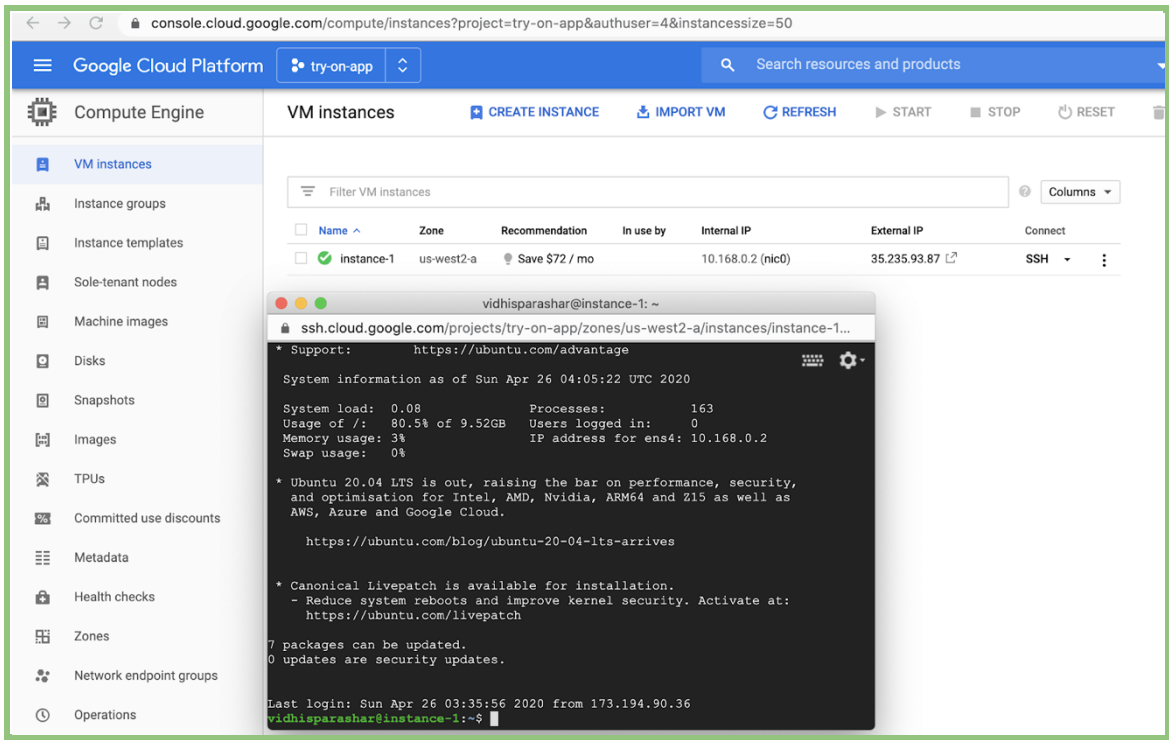

Figure 28 SSH into instance

Setup the Web App in GCE: We started with the setup of the front-end first. Since it is a node.js application, we started by installing basic node and npm libraries through apt-get utility over the Ubuntu server. We then cloned our code from the git repository using:

- *git clone https://github.com/vidhishah22/Master-Project-295A-B.git*

After cloning the application, go inside the front-end folder and run 'npm i' to install all the dependencies for the node.js project. Also, generate the privatekey.pem and certificate.pem files using the following commands:

- *openssl genrsa -out privatekey.pem 1024*
- *openssl req -new -key privatekey.pem -out certrequest.csr*
- *openssl x509 -req -in certrequest.csr -signkey privatekey.pem -out certificate.pem*

To set up the back-end, go to the back-end folder and start by creating a virtual environment. Follow below commands to do so:

- *apt-get update -y*
- *apt-get install -y python3-venv*
- *python3 -m venv venv*
- *source venv/bin/activate*

Once venv is activated, run below command to install all the back-end dependencies within python project for clothes try-on module

- *pip install -r requirements.txt*

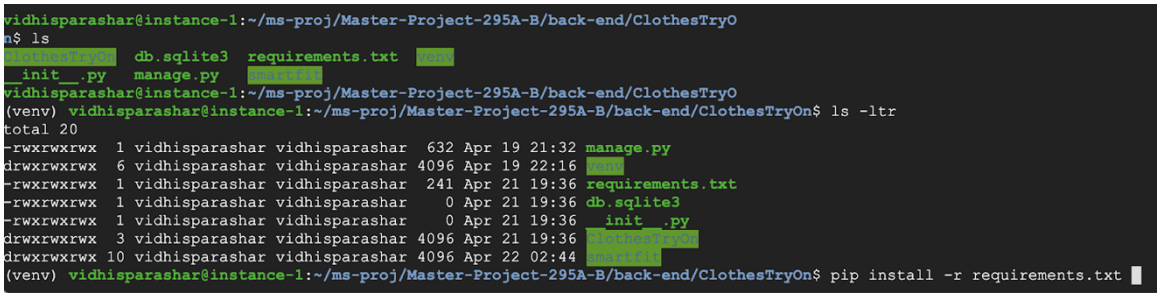

Figure 29 pip install requirements.txt

Additionally, clothes-try-on module also requires a octave library which can be installed by following below steps:

- *sudo apt-get install octave*

Once installed, invoke octave by typing:

- *octave*

Within the octave shell, install the image package with:

- *pkg install -forge package_name*
- *pkg load package_name*

Run the Web App in GCE: To start running the application, SSH into three different terminals from the cloud instance. In the first terminal, start the back-end server using the below command. Here, '&' will keep the process run as a background run

- *python manage.py runserver &*

In the second terminal, go to the front-end folder, and start running the decoder script responsible for the conversion of base 64 image into normal image fed into clothes-try on

- *./decoder.sh &*

Lastly, in the third terminal, start the node.js server with:

- npm start &

This will deploy the front-end web application over 7000 port, having a backend running on port number 8000. The website is currently up and running at: **https://35.235.93.87:7000/**

Chapter 6. Testing and Verification

Test Strategy

Introduction

During the course of this project, we performed functional testing for each model and integration testing for the whole application as well as End to end testing after deployment. The following test cases were done to test the functionality of each module that was implemented in our application.

Risks and Mitigation

As we have developed these features it is more of a White Box testing than an ideal Black Box testing, which is recommended in a testing process. White boxing testing is usually performed by the development team where the working of the code is well known to the testers. We tried to mitigate this risk by testing the features of each other rather than testing our own features.

Entry Criteria

- The user must have a browser that is supported
- The user must have a camera for real time video
- The user must grant access to the camera feed

Exit Criteria

The project will be considered a success when all three features are working together seamlessly, and users can try on all three features.

Happy Path Testing

The focus of Happy Path Testing is to test an application successfully on a positive flow. We tested this flow by making sure that the customer was able to switch on their video, pick a product and was able to view themselves trying on each of the features as expected.

Glasses:

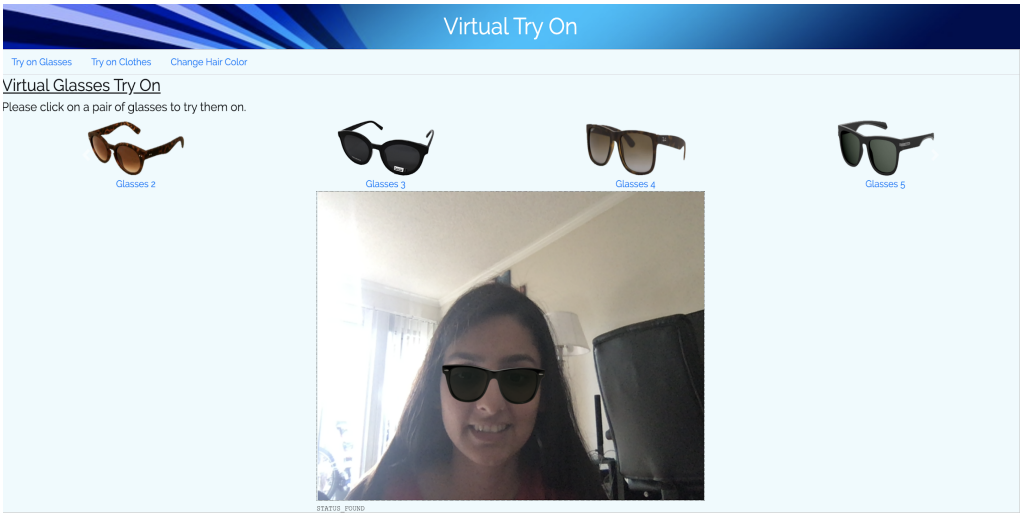

Figure 30 Glasses Tryon Tab

Hair:

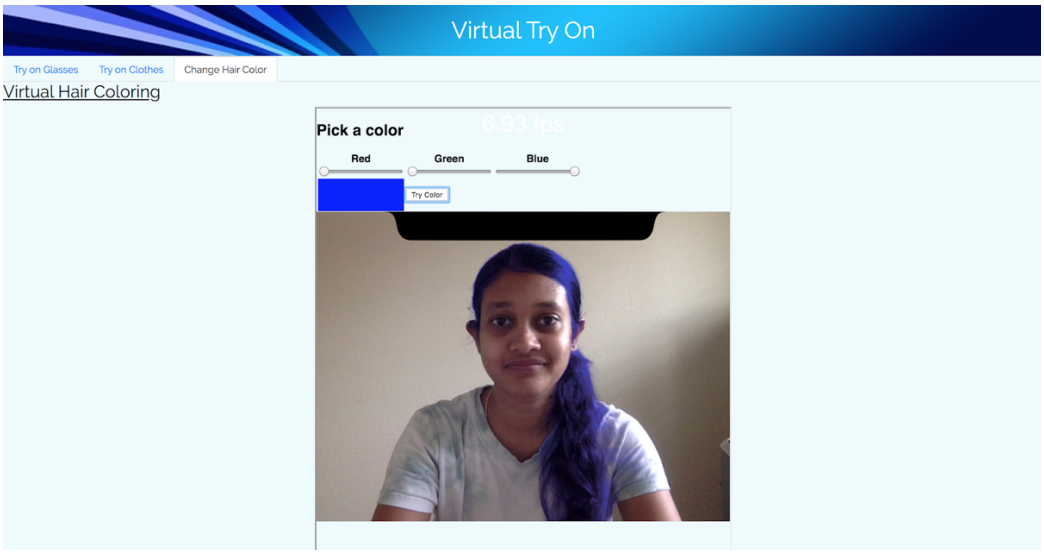

Figure 31 Hair Color Tab

Clothes:

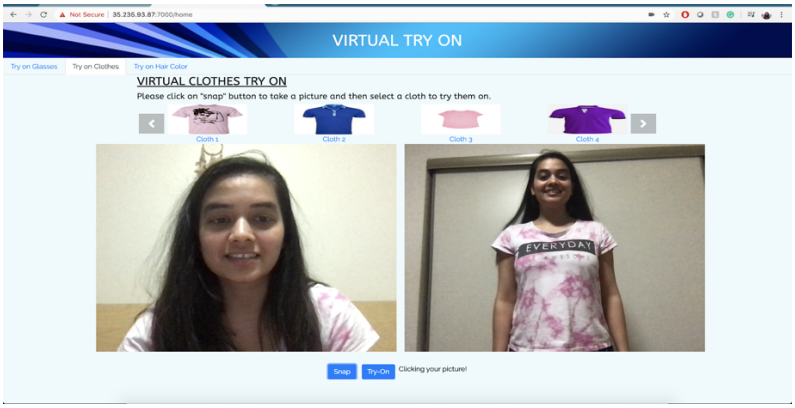

Figure 32 Clothes Tryon Tab

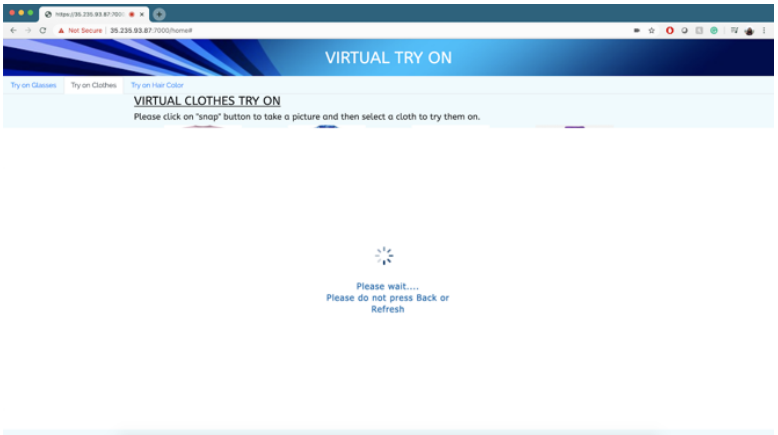

Figure 33 Processing of backend model

Figure 34 Final image with T-shirt tried on

Functional Test

Functional tests can be performed both manually and via automation, for this project we have conducted manual functional testing to make sure that all features are working without any issues.

General

Table 1 Validate that the camera requests access

Test Name	Validate that the camera requests access
Description	Once the user clicks "Try now" on the homepage, it reroutes to the tryon page and user is requested to grant access to camera
Steps	1. Start the application 2. Click "Try Now" on homepage 3. Page should reroute to try-on page 4. Browser should prompt user to give access to camera
Expected outcome	Prompt on browser for camera access

Table 2 Validate that the tabs functionality is working as required

Test Name	Validate that the tabs functionality is working as required
Description	When clicking on different tabs they should highlight and the respective content should show
Steps	1. Start the application 2. Click "Try Now" on homepage 3. Page should reroute to try-on page 4. Browser should prompt user to give access to camera

	5. Click on the “
Expected outcome	Prompt on browser for camera access

Glasses Try On

Table 3 Validate that the default tab is Glasses Try on

Test Case Name	Validate that the default tab is Glasses Try on
Description	Once the application starts the user should be rerouted to the Glasses tab from the homepage.
Steps	1. Start the application 2. Click “Try Now” on homepage 3. Page should reroute to try-on page 4. By default “Glasses Try on” page should be selected
Expected outcome	After clicking on “Try now” on homepage, user is redirected to “Glasses Try On” tab on try-on page

Table 4 Validate that default Glasses are being overlaid

Test Case Name	Validate that default Glasses are being overlaid
Description	Once the user is redirected to the try on page, the user can see the video feed with the default glass tried on.
Steps	1. Start the application 2. Click “Try Now” on homepage 3. Page should reroute to try-on page 4. By default “Glasses Try on” page should be selected 5. Default glasses should be overlaid on user
Expected outcome	On choosing a glass, the user should see how its look like on his/her successfully

Table 5 Validate that user can switch between glasses successfully

Test Case Name	Validate that user can switch between glasses successfully
Description	When the user clicks on different glasses to try on, they should be over laid correctly
Steps	1. Start the application 2. Click “Try Now” on homepage 3. Page should reroute to try-on page 4. By default “Glasses Try on” page should be selected 5. Default glasses should be overlaid on user 6. Click on different glasses options and make sure that they are overlaid correctly
Expected outcome	Switching between different glasses should be a seamless process and the glasses should be overlaid successfully

Hair Color

Table 6 Validate that the default hair color shows on user

Test Case Name	Validate that the default hair color shows on user
Description	Once the user switches to hair color tab default hair color should show.
Steps	1. Start the application 2. Click “Try Now” on homepage 3. Page should reroute to try-on page 4. By default “Glasses Try on” page should be selected 5. Click on “Hair Color” tab
Expected outcome	After clicking on “Hair Color” tab users live feed should be visible with default hair color

Table 7 Validate that the hair color changes successfully

Test Case Name	Validate that the hair color changes successfully
Description	Once the user selects a hair color that hair should show.
Steps	1. Start the application 2. Click “Try Now” on homepage 3. Page should reroute to try-on page 4. By default “Glasses Try on” page should be selected 5. Click on “Hair Color” tab 6. Scroll bars to choose hair color 7. Select “Try Color”
Expected outcome	After clicking on “Try Color” the users live feed should be visible with selected hair color

Clothes Try On

Table 8 Validate that the live feed of user is shown when tab is active

Test Case Name	Validate that the live feed of user is shown when tab is active
Description	Once the user switches to “Clothes Try on” tab, the user's video should show.
Steps	1. Start the application 2. Click “Try Now” on homepage 3. Page should reroute to try-on page 4. By default, “Glasses Try on” page should be selected 5. Click on “Clothes Try on” tab
Expected outcome	After clicking on “Clothes Try on” tab users live feed should be visible

Table 9 Validate that the image is uploaded, and selected clothes are overlaid on the user’s image.

Test Name	Validate that the image is uploaded, and selected clothes are overlaid on the user’s image.
Description	Once the user uploads their image and selects clothes to try-on, image is returned with the clothes overlaid.
Steps	6. Start the application 7. Click “Try Now” on homepage 8. Page should reroute to try-on page 9. By default, “Glasses Try on” page should be selected 10. Click on “Clothes Try on” tab 11. Upload image of user
Expected outcome	User image with selected clothes overlaid is returned.

Integration Test

Integration testing is testing of the combined functionality for all integrated modules. Integration tests usually involve a lot of code and produce traces that are larger than those produced by unit tests.

As part of this project, the main focus was to test that the backend code and the frontend UI for each feature was integrated and working as expected.

Chapter 7. Performance and Benchmarks

Performance & Benchmarks

Hair Segmentation: To quantify the performance of the segmentation models, the Intersection over Union (IoU) metric is used.

Table 10 Model Performance

Model (input)	IOU %
Full size (512×512)	81.0%
Small size (256×256)	80.2%

Hence, for the hair segmentation model also, the overlap between the target mask and prediction output was measured for a full size and small size images which got results of 81.0% and 80.2% respectively. The IoU is also referred to as Jaccard index, commonly.

Clothes try-on: The pre-trained we have used for Clothes try-on is for Pose Estimation. The model we have used was when the loss of the model was at minimal, hence the model was at its optimal position. This means that the model will give the overlaid image of the user with the best performance.

Also, as can be seen by the graph, the loss has not completely reached zero, this shows that the model was not overfit and will perform well with new clothing items as well, on which it was not trained.

Figure 35 TensorBoard Graph Visualization

Chapter 8. Deployment, Operations, Maintenance

Maintenance strategies

By using Google compute engine's powerful virtual machine, in our project, we are directly pushing any code changes into the GitHub. Hence, doing deployment multiple times is not a pain at all since it only requires the pulling the latest code from Git and re-starting the front-end and the back-end servers.

Scaling Required

Since we knew that our web application is set up on Google Compute Engine, we also planned the scaling strategies in a scenario where our website becomes really popular and gets the 100s to millions of users. In such cases, the application should be able to handle the sudden rise and fall in the traffic. First way is to handle this is a vertical scaling approach by adding more CPUs and memory to the same instance.

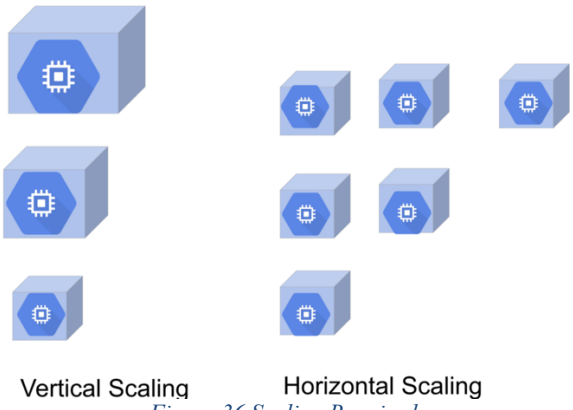

Figure 36 Scaling Required

This approach is limited based on the single machine capacity and size. On the contrary, horizontal scaling is a better option for high availability applications. As the demand increases, it allows the number of compute resources to be added dynamically.

Chapter 9. Summary, Conclusions, and Recommendations

Summary

In this project, we aimed to gain deeper knowledge and implement virtual try on for a catalogue of products. The three products focused on were glasses, clothing and hair coloring using pre-trained models, such as, CLMtrackr, pose estimation and hair segmentation, respectively. We also used libraries such as three.js and WebGL for the 3D rendering and overlaying of the products on the user's real-time video feed.

The final application was created using node.js and has been hosted on Google Cloud Platform. We have used a load balancer to make sure that the traffic is not overloading a single instance. We have created 2 instances in the US-West region.

Conclusions

We have created a client-side application running directly on any browser to support a catalogue of products to try on virtually. The application allows the user to try on glasses, clothes and change their hair color. This application could be used in the e-commerce industry to support customers in buying items which they can try on virtually from any location. As this is a virtual application on the client side, no data is collected and hence the privacy of the user is preserved. By using this application, the customer does not have to go through the hassle of spending time on driving to the store to first try-on the items they would like to purchase or return items that do not appeal to them because they can try them on virtually.

Recommendations for Further Research

At the moment the application allows for a single user experience for glasses, however, we would like to enrich this experience to allow for multiple users to try them on, as implemented for hair color.

For Clothes try on, the application currently supports the try on via an image upload with then overlays the clothing. As a future enhancement we would like to support a real-time video experience, as done in Glasses try on and hair color changing.

The hair color section of the application currently satisfies the condition of coloring the hair, however, if this application would be used in an industry such as salons, we would like to also support the changing of hair styles to give the user a more complete experience.

Glossary

Term	Definition
GAN - Generative Adversarial Network	GANs are basically made up of a system of two competing neural network models which compete with each other and are able to analyze, capture and copy the variations within a dataset.
TP-GAN	Two Path Generative Adversarial Network
WGAN	Wasserstein Generative Adversarial Network
MVC - Model View Controller	Model–View–Controller is a software design pattern commonly used for developing user interfaces which divides the related program logic into three interconnected elements.
AWS - Amazon Web Services	Amazon Web Services is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis.
E-Commerce	Electronic Commerce means buying and selling of goods, products, or services over the internet.
Real Time	To say something takes place in real-time is the same as saying it is happening "live" or "on-the-fly."
Augmented Reality	Augmented reality is an interactive experience of a real-world environment where the objects that reside in the real world are enhanced by computer-generated perceptual information
Virtual Try-On	Virtual try-on concept allows Internet visitor to “try-on“ product on website page
Generator	The generator part of a GAN learns to create

	fake data by incorporating feedback from the discriminator. It learns to make the discriminator classify its output as real.
Discriminator	The discriminator in a GAN is simply a classifier. It tries to distinguish real data from the data created by the generator.
GCP - Google Cloud Platform	Google Cloud Platform is a suite of public cloud computing services offered by Google. The platform includes a range of hosted services for compute, storage and application development that run on Google hardware.

References

1. Wang, K., Gou, C., Duan, Y., Lin, Y., Zheng, X., & Wang, F.-Y. (2017). Generative adversarial networks: introduction and outlook. *IEEE/CAA Journal of Automatica Sinica,* 4(4), 588–598. doi: 10.1109/jas.2017.7510583
2. Foster, D. (2019). Generative deep learning: teaching machines to paint, write, compose, and play. Sebastopol, CA: OReilly.
3. Olivas, E. S. (2010). Handbook of research on machine learning applications and trends: algorithms, methods, and techniques. Hershey, PA: Information Science Reference.
4. Tkachenka, A., Karpiak, G., Vakunov, A., Kartynnik, Y., Ablavatski, A., Bazarevsky, V., & Pisarchyk, S. (2019). *Real-time Hair Segmentation and Recoloring on Mobile GPUs*. ArXiv.org, ArXiv.org, July 15, 2019.
5. Mediapipe. Retrieved April 8, 2020, from https://mediapipe.readthedocs.io/en/latest/index.html
6. O'Hear, S. (2019, January 30). Wanna Kicks, a new AR app from Wannaby, lets you virtually 'try on' your next pair of kicks. Retrieved September 26, 2019, from https://techcrunch.com/2019/01/30/wanna-kicks-a-new-ar-app-from-wannaby-lets-you-virtually-try-on-your-next-pair-of-kicks/
7. Liz (2018, March 16). Feedbacks on Jeeliz Sunglasses. Retrieved September 22, 2019, from https://jeeliz.com/blog/virtual-try-on-solution-for-eyeglasses/
8. Hyprsense. (2018, May 29). Lologem's Virtual Jewelry Try-on Service. Retrieved September 26, 2019, from https://medium.com/@hyprsense/lologems-virtual-jewelry-try-on-service-b339c17a3a4b

9. Q-Success. Usage statistics of JavaScript as client-side programming language on websites. Retrieved April 8, 2020, from https://w3techs.com/technologies/details/cp-javascript/

10. Q-Success. Usage statistics of JavaScript libraries for websites. Retrieved April 8, 2020, from https://w3techs.com/technologies/overview/javascript_library

11. The Jquery Foundation. jQuery: The write less, do more, JavaScript library. Retrieved April 8, 2020, from https://jquery.com/

12. Tutorialspoint. Retrieved on April 8, 2020, from https://www.tutorialspoint.com/unix/shell_scripting.htm

13. Parisi, T. (2012). WebGL: up and running. Sebastopol, CA: OReilly.

14. RohanBhandari. (n.d.) RohanBhandari/keras_Realtime_Multi-Person_Pose_Estimation. Retrieved from https://github.com/RohanBhandari/keras_Realtime_Multi-Person_Pose_Estimation

15. ZheC. (2020, February 17). ZheC/Realtime_Multi-Person_Pose_Estimation. Retrieved from https://github.com/ZheC/Realtime_Multi-Person_Pose_Estimation

16. Liang, X., Gong, K., Shen, X., & Lin, L. (2019). Look into Person: Joint Body Parsing & Pose Estimation Network and a New Benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(4), 871–885. doi: 10.1109/tpami.2018.2820063

17. Gong, K., Liang, X., Zhang, D., Shen, X., & Lin, L. (2017). Look Into Person: Self-Supervised Structure-Sensitive Learning and a New Benchmark for Human Parsing. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

18. Han, X., Wu, Z., Wu, Z., Yu, R., & Davis, L. S. (2018). VITON: An Image-Based Virtual Try-on Network. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. doi: 10.1109/cvpr.2018.00787

19. Cao, Z., Simon, T., Wei, S.-E., & Sheikh, Y. (2017). Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi: 10.1109/cvpr.2017.143

20. Wei, S.-E., Ramakrishna, V., Kanade, T., & Sheikh, Y. (2016). Convolutional Pose Machines. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi: 10.1109/cvpr.2016.511

21. Serving websites | Solutions | Google Cloud. (n.d.). Retrieved from https://cloud.google.com/solutions/web-serving-overview

22. Building a Node.js app on App Engine. (n.d.). Retrieved from https://cloud.google.com/appengine/docs/standard/nodejs/building-app

23. Vergadia, P. (2019, September 27). Hosting a website on Google Cloud using Google Compute Engine. Retrieved from https://medium.com/google-cloud/hosting-a-website-on-google-cloud-using-google-compute-engine-c6fe84d76f51

24. John, R. T. (2018, July 20). Introduction to Compute Engines on GCP. Retrieved from https://medium.com/@robertjohn_15390/introduction-to-compute-engines-on-gcp-238e012cc2eb

25. Saragih, J. M., Lucey, S., & Cohn, J. F. (2009). Face alignment through subspace constrained mean-shifts. 2009 IEEE 12th International Conference on Computer Vision. doi: 10.1109/iccv.2009.5459377