CBS 1901 TARP Project

Title: Virtual Fit to predict the fitness of the dress using DL/ML

Team Members:

Aditya Malpani (20BBS0062)

Ayush Garg (20BBS0109)

Ayan Pandey (20BBS0173)

Avantika Mapari (20BBS0075)

Under the guidance of:

Professor BALAKRISHNAN P

Abstract:

Customers can digitally try on clothes using virtual try-on clothing software, which eliminates the need to physically try on items or visit a store. The software creates a realistic image of the user's body using cutting-edge 3D modeling and computer vision technologies, enabling users to visualize how various clothing items might appear on them.

Customers can simply upload a photo of themselves or take a picture with a camera to use the virtual try-on clothing software. The program then analyses the photograph using complex algorithms to produce a 3D representation of the user's physique. In order to provide a very accurate depiction of the user, the program can take into account the user's height, weight, and body form in addition to other elements like skin tone and hair color.

The user can browse a virtual catalog of apparel after the 3D model has been built and choose the items they want to try on. The software then superimposes the piece of clothing onto the user's 3D body model so they can view how it would appear on them from every angle.

The way individuals shop for clothing may change as a result of virtual try-on software. Customers may now more easily and conveniently find the ideal fit and style without having to leave their homes. Because shoppers may try on items before they buy them, it can also assist in decreasing the frequency of returns and exchanges.

The fashion business can potentially benefit from virtual try-on software. It enables creators and merchants to present their goods in a more interactive and interesting way, which may boost sales and brand recognition. In order to improve product design and inventory control, it also offers useful information on customer preferences and body dimensions.

Overall, virtual try-on clothes software is an exciting and innovative technology with significant potential to transform the fashion industry and improve the shopping experience for customers.

Background and Salient Features:

The concept of Virtual Try-On has gained traction due to the growing popularity of online shopping and the challenges associated with buying clothes without the opportunity to physically try them on. VTO technology bridges this gap by utilizing advanced computer vision, image processing, and AR/VR techniques to create a digital representation of how clothing would appear on a user's body.

Virtual try-on clothes software is a technology that allows customers to try on clothes virtually, which has been rapidly evolving in recent years. The software uses advanced 3D modeling and computer vision technology to create a realistic representation of the user's body, allowing them to see how different clothing items would look on them without physically trying them on.

One type of virtual try-on clothes software involves using augmented reality (AR) technology. This involves overlaying a digital image of the clothing item onto the user's live camera feed, allowing them to see themselves wearing the item in real-time. Another type of virtual try-on software uses computer-generated models that are customized to the user's body shape and size. The user can then browse a virtual catalog of clothing items and see how each item would look on their customized model.

The main objective of the proposed system is to enhance the customer experience in clothing fitting by enabling customers to virtually try clothing on in order to check for size, fit, or style. The main **features** of our product are mentioned below:

1. To detect and extract human body skeleton-based joint positions using a web camera.
2. To calculate body measurements based on the extracted body skeleton joint positions.
3. To fit virtual garments onto the human body according to the extracted body skeleton joint positions, body measurements, and garment measurements.

Related Works:

S.No	Citation	Advantages	Disadvantages
1	Minar, M. R., Tuan, T. T., Ahn, H., Rosin, P., & Lai, Y. K. (2020, June). Cp-vton+: Clothing shape and texture preserving image-based virtual try-on. In *CVPR Workshops* (Vol. 3, pp. 10-14).	Focuses on preserving both the shape and texture of the clothing during the virtual try-on process. The method is image-based, which means it does not require 3D models or complex garment parameterization. This makes it relatively easier to implement and use in practical applications. The use of GANs (Generative Adversarial Networks) allows the model to generate clothing images that are coherent with the target person's appearance.	GANs tend to be complex and computationally expensive. Implementing and training such models might require significant computational resources and expertise. The quality of generated images in GAN-based approaches can be inconsistent. The paper does not discuss in detail how the proposed method's results are quantitatively evaluated. The success of the virtual try-on process might heavily depend on the quality of the input images (both the source clothing and the target person).

2	Esser, P., Sutter, E., & Ommer, B. A Variational U-Net for Conditional Appearance and Shape Generation.	Variational methods combine the benefits of generative models with probabilistic modelling, allowing for the generation of diverse and realistic samples while also providing a measure of uncertainty in the generated outputs. The U-Net architecture is known for its effectiveness in image-to-image tasks, particularly in tasks involving image segmentation, translation, and generation. The paper's focus on conditional appearance and shape generation suggests the potential to control and manipulate the appearance and shape of generated outputs, which could be valuable in applications like image editing and synthesis.	Variational methods can introduce complexity due to the probabilistic modelling involved. Variational U-Net architectures may suffer from mode collapse, where the model generates a limited set of outputs, failing to capture the full diversity of the target distribution. If the training data is biased or lacks diversity, the model might struggle to generate accurate and diverse samples. Achieving a balance between controlling appearance and shape while maintaining realistic and coherent outputs can be difficult, especially when generating complex or detailed images.

3	Siarohin, A., Sangineto, E., Lathuiliere, S., & Sebe, N. (2018). Deformable GANs for pose-based human image generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3408-3416).	The approach allows for generating human images in a wide range of poses. This is useful in applications such as virtual avatars, animation, and clothing design, where the ability to generate images with varying poses is essential. Deformable GANs aim to generate images that are visually realistic and consistent with human anatomy and pose variations. This can enhance the quality of synthetic images compared to traditional methods. Instead of relying solely on real-world images for each pose, this approach can generate new images in different poses, potentially reducing the need for extensive pose-specific training data.	Deformable GANs might produce artifacts, distortions, or unrealistic body parts in some generated images. raining Deformable GANs can be complex and resource-intensive. It may require careful tuning of hyperparameters, significant computational resources, and a large amount of training data. While the model might perform well on the poses present in the training data, its ability to generalize to unseen poses or extreme pose variations might be limited.

4	Ge, C., Song, Y., Ge, Y., Yang, H., Liu, W., & Luo, P. (2021). Disentangled cycle consistency for highly-realistic virtual try-on. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 16928-16937).	Disentangled cycle consistency is designed to improve the realism of virtual try-on. The approach aims to disentangle various factors such as clothing appearance, body shape, and pose. This disentanglement could result in greater flexibility and control over how clothing items are applied to different body types and poses. By focusing on cycle consistency, the proposed approach might mitigate artifacts and inconsistencies that can occur in virtual try-on systems, leading to a higher quality output.	Depending on the approach's complexity, there could be a risk of overfitting to the training data, leading to poor generalization to new data and scenarios. Disentangling factors in virtual try-on can introduce computational and technical complexities. While the proposed approach might work well under controlled conditions, its performance in real-world scenarios with diverse clothing styles, body shapes, and lighting conditions could be challenging.

5	Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training gans. *Advances in neural information processing systems*, *29*.	The paper proposes techniques like using minibatch discrimination and historical averaging of the generator weights. These techniques can help stabilize GAN training by preventing mode collapse and improving convergence. The stabilization techniques and insights provided in the paper can potentially lead to faster convergence during training, reducing the overall time required to train GANs effectively. The techniques introduced in the paper are not specific to a certain type of GAN architecture or application, making them broadly applicable across various domains and use cases.	While the paper provides valuable insights, there is no guarantee that the proposed techniques will work optimally for all types of GAN architectures and tasks. GAN training is known to be sensitive to hyperparameters, and the techniques proposed in the paper might also require tuning specific hyperparameters to achieve optimal results.

6	Xi Chen, Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In NIPS, pages 2172–2180, 2016.	InfoGAN focuses on learning representations that capture specific factors of variation in the data. This can result in more interpretable and semantically meaningful representations that correspond to attributes like pose, orientation, etc. The InfoGAN architecture encourages the disentanglement of latent factors, meaning that different components of the latent space correspond to separate and independent features of the data. Can be extended to incorporate supervised learning by mapping known labels to interpretable latent codes.	Training InfoGAN and similar models can be more complex compared to standard GANs, as they involve multiple components such as the generator, discriminator, and mutual information maximization. While InfoGAN demonstrates promising results for certain types of data and attributes, its effectiveness may vary across different domains. As with many GAN-based models, there is a risk of mode collapse, where the generator focuses on a limited set of generated outputs, reducing diversity.

7	Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In CVPR, pages 1096–1104, 2016.	The DeepFashion dataset is known for its rich annotations, including attributes, landmarks, and bounding boxes, making it suitable for various tasks like clothes recognition, retrieval, and attribute prediction. The dataset addresses the challenges of recognizing and retrieving fashion items in images, which has applications in e-commerce, fashion recommendation, and virtual try-on systems. The dataset is based on real-world images, which enhances the practicality of the research conducted using it	The quality of the dataset is dependent on the quality of the source images, which could include issues like variations in lighting, resolution, and image quality. While diverse, the dataset might still have limitations in terms of capturing all possible fashion styles, cultures, and demographics, potentially limiting the generalization of models trained on it. Like many real-world datasets, DeepFashion might suffer from class imbalance, where certain clothing items or attributes are underrepresented, potentially affecting model performance.

8	Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. arXiv, 1605.02226, 2016.	The proposed method offers a way to model complex and high-dimensional probability distributions using neural networks, enabling the modelling of intricate data distributions. The autoregressive approach can be used for generative modelling, allowing the generation of new data samples from the learned distribution. Neural networks can capture intricate patterns in data and can scale to large and high-dimensional datasets, making them suitable for various data modelling tasks.	Autoregressive models might struggle to capture correlations between distant data components, as each component's prediction depends only on the previous components. Modelling high-dimensional data with neural networks can be challenging due to the curse of dimensionality and the need for substantial training data. Autoregressive models might struggle to capture correlations between distant data components, as each component's prediction depends only on the previous components.

9	Zhang, M., Lin, L., Pan, Z., & Xiang, N. (2015). Topology-independent 3D garment fitting for virtual clothing. *Multimedia Tools and Applications*, *74*, 3137-3153.	The paper focuses on topology-independent fitting, which means that the approach aims to fit garments to 3D human models regardless of the specific topology or mesh structure of the models. This can lead to more versatile and flexible garment fitting methods. Topology-independent 3D garment fitting has practical applications in the fashion industry, enabling designers and retailers to visualize and showcase clothing items on virtual models. The approach could potentially be extended to allow customization and personalization of virtual clothing, tailoring the fit to individual preferences and body shapes.	Topology-independent garment fitting might require significant computational resources, especially if it involves simulations or optimizations to achieve accurate results. High-quality 3D human models and detailed clothing models are needed to achieve accurate fitting. Achieving realistic 3D garment fitting is a complex task that involves considerations such as fabric behavior, body dynamics, and user interactions.

10	Mir, A., Alldieck, T., & Pons-Moll, G. (2020). Learning to transfer texture from clothing images to 3d humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7023-7034).	The paper addresses the challenge of transferring textures from clothing images to 3D human models, contributing to more realistic and visually appealing virtual clothing. Texture transfer can enable designers to experiment with creative textures, patterns, and styles in a virtual environment, facilitating innovation in fashion design. Manually creating textures for 3D human models can be time-consuming. Automated texture transfer methods can reduce the effort required for texture generation.	Texture transfer involves complex challenges, such as handling viewpoint changes, lighting variations, and deformation. The success of texture transfer depends on the quality of the input clothing images and the accuracy of the 3D human models. Poor-quality textures or models can result in unrealistic or undesirable outcomes.

11	Meng, Y., Mok, P. Y., & Jin, X. (2012). Computer aided clothing pattern design with 3D editing and pattern alteration. Computer-Aided Design, 44(8), 721-734.	Computer-aided design tools can streamline the clothing pattern design process, making it more efficient and less labor-intensive compared to traditional manual methods. Incorporating 3D editing in pattern design can provide a more realistic and intuitive way to visualize and manipulate clothing patterns, allowing designers to see the impact of alterations in a 3D context. The ability to alter clothing patterns using computer tools enables designers to experiment with different designs, sizes, and styles without the need for physical prototypes.	Design software can have complex interfaces and features, which might be overwhelming for some users, potentially slowing down the design process initially. Achieving accurate fit and proportions in digital clothing patterns is important, but it might require additional efforts to validate against real-world measurements and bodies. Integrating computer-aided design tools into existing design workflows might require adjustments and compatibility considerations.

12	Lagė, A., & Ancutienė, K. (2019). Virtual try-on technologies in the clothing industry: basic block pattern modification. *International Journal of Clothing Science and Technology*, *31*(6), 729-740.	Applying virtual try-on technologies to basic block patterns can streamline the process of pattern modification, allowing designers to quickly visualize and adjust patterns. Can potentially reduce the need for physical prototypes, saving time and costs associated with materials and sample creation.	Over-reliance on basic block pattern modification might not account for individual body shapes and variations. This can result in fit issue Basic block patterns are simplified representations of garments. Modifying these patterns might not accurately capture the intricate details and complexities of certain designs.
13	Kuppa, G., Jong, A., Liu, X., Liu, Z., & Moh, T. S. (2021). ShineOn: Illuminating design choices for practical video-based virtual clothing try-on. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 191-200).	The paper addresses the challenge of practical video-based virtual clothing try-on, which involves real-time or near-real-time visualization of clothing on video streams. ShineOn aims to provide more realistic visualizations of virtual clothing on videos, potentially enhancing user engagement and satisfaction. The method's focus on illuminating design	Achieving real-time or near-real-time video-based virtual try-on involves complex computational and rendering requirements, which might challenge hardware capabilities and affect user experience. Effective video-based virtual try-on requires accurate 3D models, detailed clothing textures, and realistic fabric simulations, which might be resource-intensive to obtain or create.

		choices implies that it helps designers make informed decisions to optimize the virtual try-on process	Integrating practical video-based virtual try-on into e-commerce platforms and other applications might require compatibility and integration challenges.
14	Magnenat-Thalmann, N., Kevelham, B., Volino, P., Kasap, M., & Lyard, E. (2011). 3d web-based virtual try on of physically simulated clothes. *Computer-Aided Design and Applications*, *8*(2), 163-174.	Improves upon existing VTO solutions by providing both animation of the avatar and realistic physical simulation of the 3D garments. Within the application we always maintain a direct connection to the underlying CAD data, we envision exporting the results of our interactive 3D configuration to 2D CAD data that can be fed into fabric cutters, pattern plotters or maybe even sewing robots to produce the 3D customized and personalized garments.	The bottlenecks are in the garment simulation and collision detection. It is not possible to obtain huge increases in performance through further optimization of classic simulation techniques.

15	Tuan, T. T., Minar, M. R., Ahn, H., & Wainwright, J. (2021). Multiple pose virtual try-on based on 3d clothing reconstruction. *IEEE Access*, *9*, 114367-114380.	The results show that a hybrid approach based on the 3D clothing model reconstruction in CloTH-VTON can generate more natural clothing deformation results for multi-posed VTON applications.	The proposed 3D-MPVTON has limitations in the 3D clothing reconstruction step and the target pose and customer shape transfer step. Despite the texture matching algorithm, the artifacts at the clothing side/boundary for some results are noticeable
16	Tsunashima, H., Arase, K., Lam, A., & Kataoka, H. (2020). Uvirt—unsupervised virtual try-on using disentangled clothing and person features. *Sensors*, *20*(19), 5647.	We can apply UVIRT to practical scenarios for virtual try-on in a real-world C2C market. Conventional supervised virtual try-on methods cannot be applied to the C2C market dataset because the accumulation of annotated data for the C2C market is very difficult.	The annotation cost of the training data for virtual try-on is high because the conventional supervised approaches need annotated data such as clothing-parsing semantic segmentation masks and paired images. Issues with distortion of the person and deficiencies in face reconstruction.

17	Shen, C., Liu, Z., Gao, X., Feng, Z., & Song, M. (2023). Self-adaptive Clothing Mapping based Virtual Try-on. *ACM Transactions on Multimedia Computing, Communications and Applications*.	The deformation module Color-adaptive Clothing Mapping achieves great results in various clothing properties and textures, with the contours of the deformed clothing fitting the model most closely. The proposed clothing mapping technique models the changes in the pixel position and the pixel value, which is expected to achieve optimal clothing deformation performance.	Like previous methods, we still do not perfectly solve some difficult cases such as long sleeves self-occlusion and complex model poses (raised arms, extreme sideways). Only considers the mapping for clothing currently, the mapping of models can also be considered to improve the quality of virtual try-on further.
18	Frâncu, M., & Moldoveanu, F. (2015). Virtual try on systems for clothes: Issues and solutions. *UPB Scientific Bulletin, Series C*, *77*(4), 31-44.	Differentiates between cloth collisions with external objects and self-collisions (including collisions with other pieces of cloth). Most straightforward way of accelerating cloth simulation and collision detection as well as the other modules of the VTO is parallelization and to do that we	Providing a real time simulation of realistic apparel on current generation computers or mobile devices is still a difficult task despite all the hardware advances. The collision detection phase is the other bottleneck besides the physical solver and it is also need of acceleration data structures and parallelization.

		developed new constraint solvers based on the Jacobi and Conjugate Residuals methods and were implemented in parallel on the GPU using OpenCL kernels.	
19	Tan, Z. L., Bai, J., Zhang, S. M., & Qin, F. W. (2021). NL-VTON: a non-local virtual try-on network with feature preserving of body and clothes. *Scientific Reports*, *11*(1), 19950.	Non-local methods can capture long-range dependencies between body and clothing features, potentially leading to more realistic virtual try-on results that align better with how clothing drapes and interacts with the body. Non-local grid regularization loss is designed and applied to the stage of cloth deformation, so as to retain the clothes' global structures in the complex clothes deformation process better.	NL-VTON cannot distinguish the front and back of the collar very well and judge whether the person is forward or backward thus, generate failed result. The existing quantitative evaluation can only test the results of wearing the same clothing. It needs to be combined with qualitative evaluation to judge whether the model is good or bad.

20	Hsieh, C. W., Chen, C. Y., Chou, C. L., Shuai, H. H., Liu, J., & Cheng, W. H. (2019, October). FashionOn: Semantic-guided image-based virtual try-on with detailed human and clothing information. In *Proceedings of the 27th ACM international conference on multimedia* (pp. 275-283).	FashionOn addresses CP-VTON's limb occlusion by concurrently warping clothing and body masks, ensuring coherence, and sequentially rendering human appearance, differing from direct TPS warp and synthesis. Generates the pleats and shadows based on the body shape and the posture of the source person which achieves far more realistic and reasonable results.	Fails to synthesize symmetric eyes in the case of transforming the sideways photo into front one since the face of the sideways photo often contains either a hidden eye or asymmetric size of eyes. The fingers of generated person images are blurry because current human parsing regards the fingers as a region, regardless of each finger.

Proposed Model (Stages of Implementation):

The approach of this project is to create a virtual trial room for humans to try various clothing virtually. For our real-time implementation, we used the open-source library, OpenCV.

- The first step is to obtain the video stream of the user with the help of a web camera. With the OpenCV, each frame is converted into a matrix in grayscale.
- Creation of cloth masks with the help of bitwise operation.
- Positioning of the cloth masks by using face detection.
- Resizing of the masks of cloths by using area interpolation.
- Superimposition of the masks of cloths on the user.

Open CV:
OpenCV stands for Open-Source Computer Vision. The library has more than 2,500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high-resolution image of an entire scene, find similar images.

Image Processing:
Image processing refers to the manipulation and analysis of digital images using various algorithms and techniques. In the context of a virtual try-on model, image processing is used to enable the model to simulate how clothing items would look on a person. This involves capturing images or videos of the clothing items and the person, processing those images to extract features like the size and shape of the clothing, and then using that information to create a simulation of how the clothing would look on the person. Image processing techniques used for virtual try-on models may include color correction, image segmentation, feature detection, and machine learning algorithms that can identify and match clothing items to a person's body shape and size.

Mask Creation:
Mask creation is a technique used in image processing to isolate or segment specific areas of an image or video. In the context of a virtual try-on model, mask creation is used to separate the clothing item from the background and the person wearing it, allowing the model to simulate how the clothing item would look on a different person or background. Thresholding is a segmentation

technique used for separating an object from its background. It involves comparing each pixel of the image with a pre-defined threshold value.

Mask Creation

Resizing and Superimposition of mask:

We have created the mask, but the mask contains some redundant data that we do not need therefore we will select the region of interest (ROI) in which we will be selecting the area in our user body and superimposing the mask on the body of the user such that it should be properly aligned with the user body.

To create a mask superimposition, the model uses the mask of the clothing item, as well as various image processing techniques such as perspective transformation and color correction, to adjust the mask to match the pose and lighting of the person in the image or video. The adjusted mask is then superimposed on top of the person, allowing the model to simulate how the new clothing item would look on them.

The overall idea of our aim output

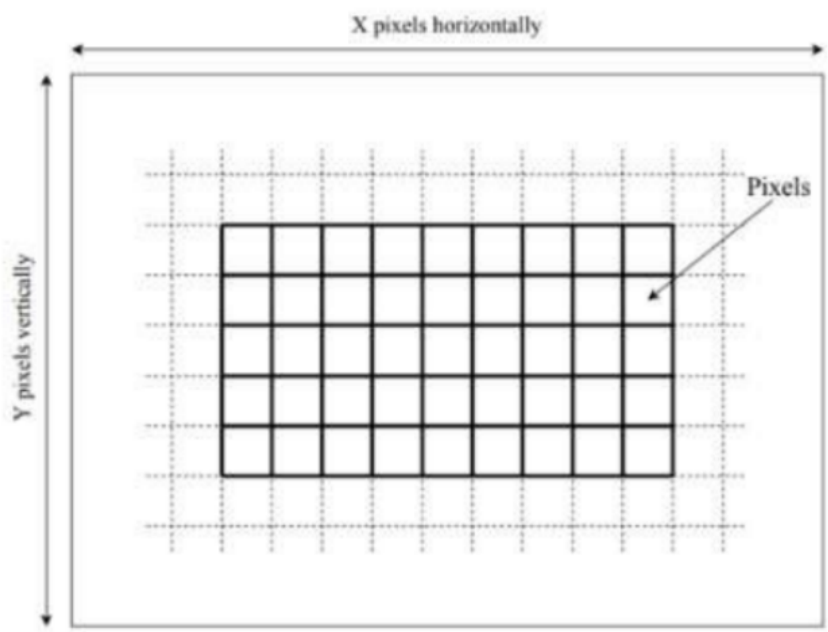
Implementation:

Stage 1: Specify and find out hardware and software requirements for the system to be able to implement the application.

Stage 2: Human Body Recognition, Detection, and Motion tracking (Using TensorFlow). The model detects 14 joint positions, including top, neck, right shoulder, right elbow, right wrist, left shoulder, left elbow, left wrist, right hip, right knee, right ankle, left hip, left knee, and left ankle. We can also employ the OpenCV library to process image data and supply it to the TensorFlow Lite pose estimation model. OpenCV will read the frame and feed image data into the pose estimation model classifier for pose estimation.
This will be followed by improving the quality of the image and the accuracy of classification.

Stage 3: Generation of garment model. In order to improve the virtual clothing fitting experience and provide accurate size and style to users, the garment model used in this project is customized to match the skeleton joint positions. The garment model is created using an actual garment with a transparent background and saved in the Portable Network Graphics (PNG) file format. This format is chosen for its ability to support lossless data compression and preserve data without any loss each time it is saved or opened. Additionally, the PNG format also supports transparency, which is a clear advantage over other file formats such as Joint Photographic Experts Group (JPEG).

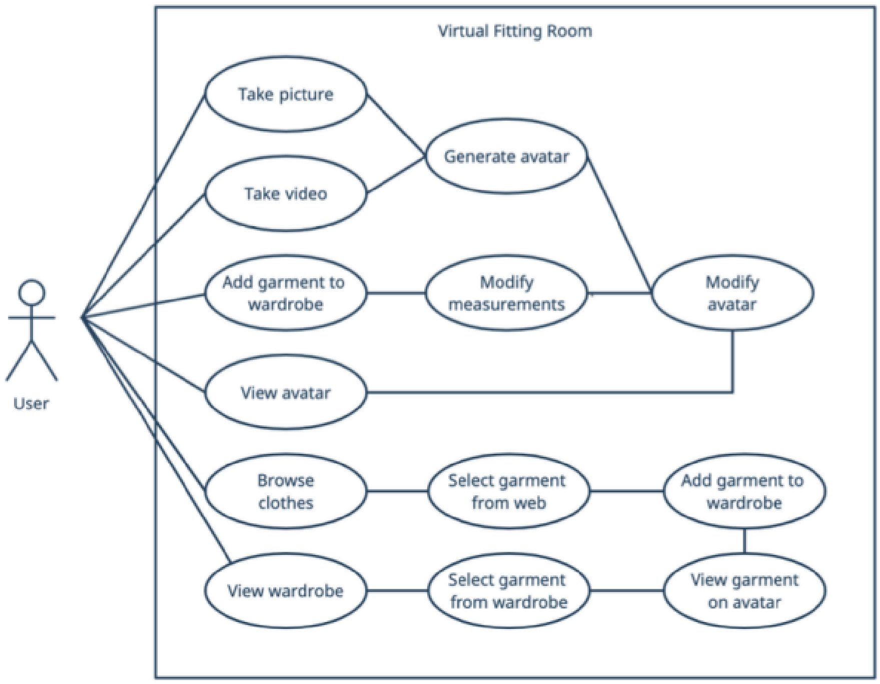
Stage 4: Superimposition of garment model over the Human Body. The virtual garment is placed over the body by utilizing data of body measurements, body skeleton joint positions, and garment measurements. To achieve this, the garment model is first customized to match the skeleton joint positions. The model is generated based on an actual garment with a transparent background. Next, the garment model is converted into a Bitmap class, which creates two-dimensional matrices composed of pixels, with each pixel containing a color value. Each pixel of the Bitmap is processed in a customized way to align with the body skeleton joint points.
We can then use android.graphics.Matrix package to perform scaling and transformation of the garment model to generate a matrix, based on the body measurements. These measurements include shoulder width, left and right arm length, left and right limb length. The scaling of the garment model is implemented based on the calculated distances between each of the body skeleton joint positions. The resulting matrix is used to place the virtual garment over the body



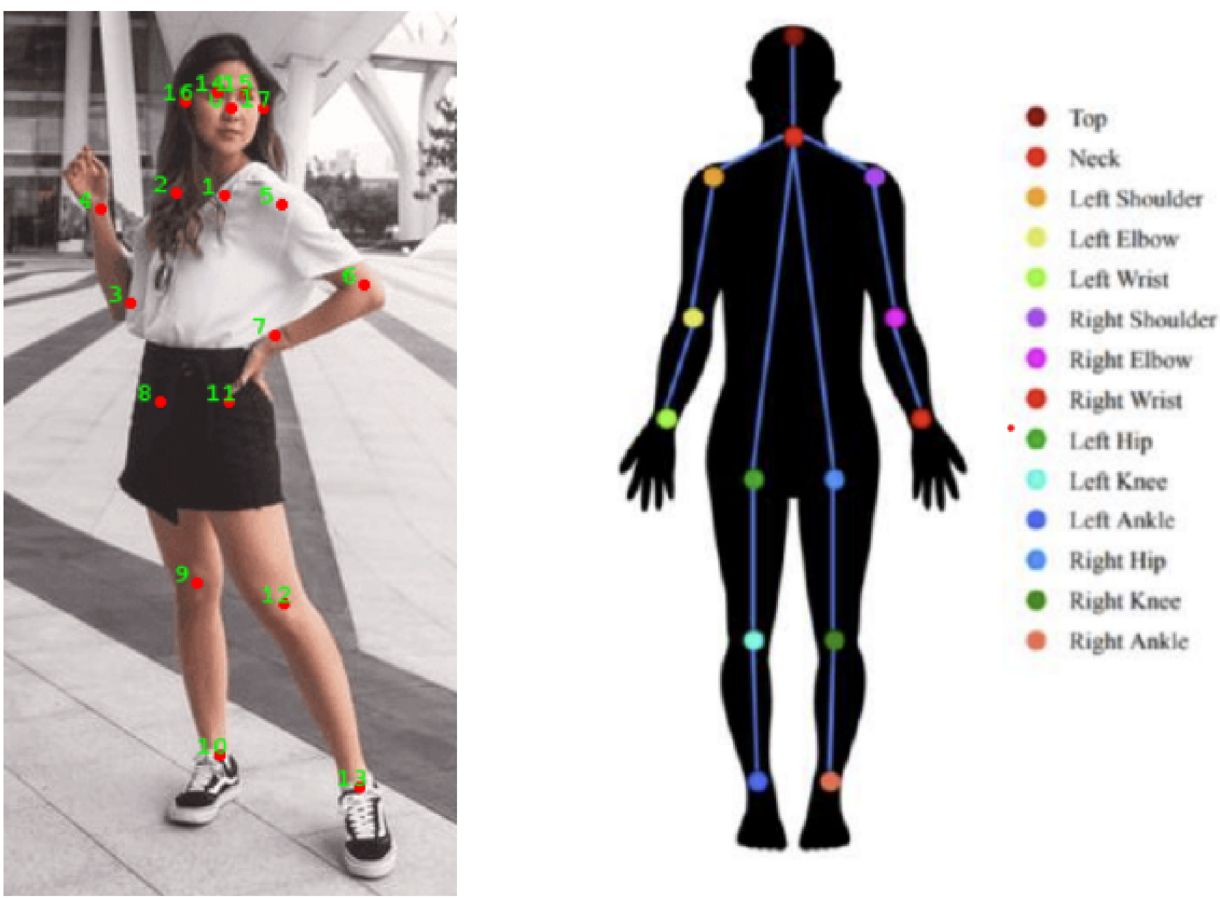
Pixel description of the Bitmap

Stage 5: Build a web server that allows users to upload images, select clothing items, and view the results of the virtual try-on process using Flask. Flask can also be used to integrate with other technologies, such as TensorFlow and OpenCV to provide a seamless user experience

Diagrammatic Representation:

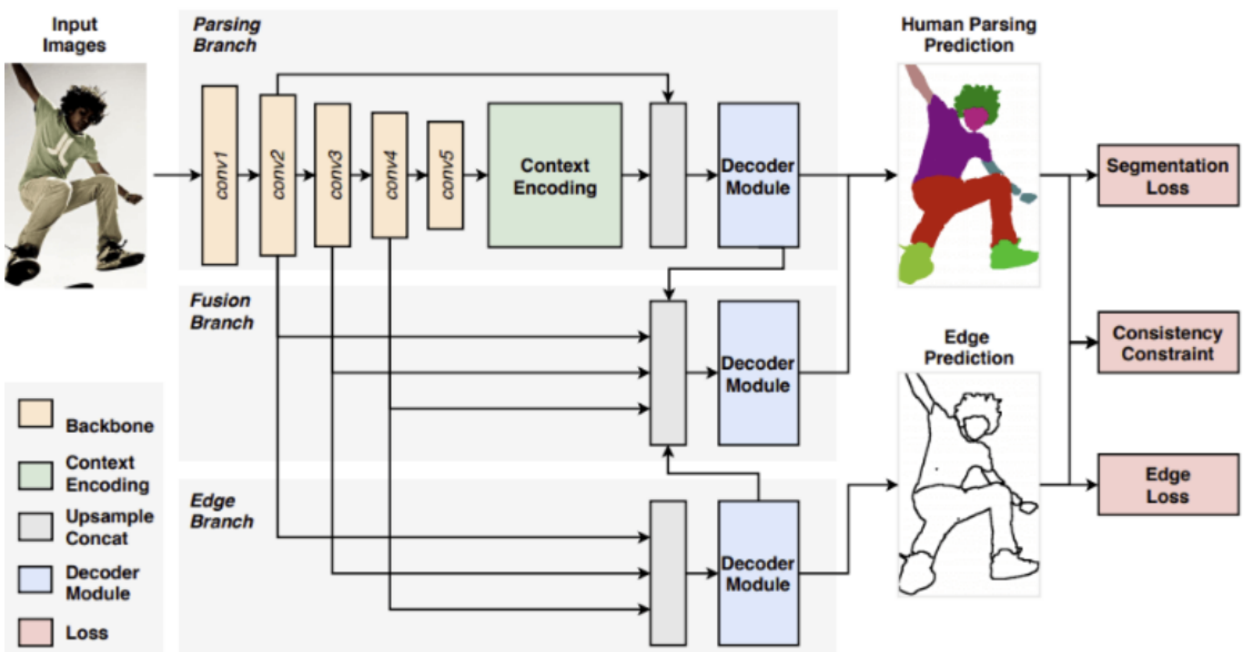


Use Case diagram for a user interacting with the virtual try-on model



Human Body Recognition, Detection and Motion Tracking

A step-by-step process of converting an image to mask



Implementation after Hardware and Software requirements are met

Implemented Code:

```
app.py  ×
app.py > predict
from flask import Flask, render_template, request
# import json
from flask_cors import CORS
# import numpy as np
import cv2                              # Library for image processing
# from math import floor
import random

app = Flask(__name__)
CORS(app)


@app.route("/")
def index():
    return render_template('index.html')


@app.route('/pant.html')
def ploty():
    return render_template('pant.html')


@app.route('/predict/', methods=['GET', 'POST'])
def predict():
    shirtno = int(request.form["shirt"])
    pantno = int(request.form["pant"])

    cv2.waitKey(1)
    cap = cv2.VideoCapture(0)
    ih = shirtno
    i = pantno
    shirtWidth = 0
    shirtHeight = 0
    pantWidth = 0
    pantHeight = 0
    while True:
        imgarr = ["shirt1.png", 'shirt2.png', 'shirt51.jpg', 'shirt6.png',]
```

```

        imgshirt = cv2.imread(imgarr[ih-1], 1)  # original img in bgr
        if ih == 3:
            shirtgray = cv2.cvtColor(imgshirt, cv2.COLOR_BGR2GRAY)
            # there may be some issues with image threshold...depending on the color/contrast of image
            ret, orig_masks_inv = cv2.threshold(
                shirtgray, 200, 255, cv2.THRESH_BINARY)
            orig_masks = cv2.bitwise_not(orig_masks_inv)

        else:
            # grayscale conversion
            shirtgray = cv2.cvtColor(imgshirt, cv2.COLOR_BGR2GRAY)
            # there may be some issues with image threshold...depending on the color/contrast of image
            ret, orig_masks = cv2.threshold(
                shirtgray, 0, 255, cv2.THRESH_BINARY)
            orig_masks_inv = cv2.bitwise_not(orig_masks)
        origshirtHeight, origshirtWidth = imgshirt.shape[:2]
        imgarr = ["pant7.jpg", 'pant21.png']
        imgpant = cv2.imread(imgarr[i-1], 1)
        imgpant = imgpant[:, :, 0:3]  # original img in bgr
```

```python
        # grayscale conversion
        pantgray = cv2.cvtColor(imgpant, cv2.COLOR_BGR2GRAY)
        if i == 1:
            # there may be some issues with image threshold...depending on the color/contrast of image
            ret, orig_mask = cv2.threshold(
                pantgray, 100, 255, cv2.THRESH_BINARY)
            orig_mask_inv = cv2.bitwise_not(orig_mask)
        else:
            ret, orig_mask = cv2.threshold(
                pantgray, 50, 255, cv2.THRESH_BINARY)
            orig_mask_inv = cv2.bitwise_not(orig_mask)
        origpantHeight, origpantWidth = imgpant.shape[:2]

        face_cascade = cv2.CascadeClassifier(
            'haarcascade_frontalface_default.xml')

        ret, img = cap.read()

        height = img.shape[0]
        width = img.shape[1]

        resizewidth = int(width*3/2)
        resizeheight = int(height*3/2)

        cv2.namedWindow("img", cv2.WINDOW_NORMAL)
        cv2.resizeWindow("img", resizewidth, resizeheight)

        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)

        for (x, y, w, h) in faces:  # strt pt  end pt
           # cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
            # cv2.rectangle(img,(x-100,y+100),(x+w-100,y+h+100),(255,255,255),2)

            pantWidth = 3 * w  # approx wrt face width
            # preserving aspect ratio of original image..
            pantHeight = pantWidth * origpantHeight / origpantWidth

            # Center the pant..just random calculations..
            if i == 1:
                x1 = x-w
                x2 = x1+3*w
                y1 = y+5*h
                y2 = y+h*10
            elif i == 2:
                x1 = x-w/2
                x2 = x1+2*w
                y1 = y+4*h
                y2 = y+h*9
            else:
                x1 = x-w/2
                x2 = x1+5*w/2
                y1 = y+5*h
                y2 = y+h*14
            # Check for clipping(whetehr x1 is coming out to be negative or not..)

            # two cases:

            """
            close to camera: image will be to big
            so face ke x+w ke niche hona chahiye warna dont render at all
            """

```

```python
            if x1 < 0:
                x1 = 0  # top left ke bahar
            if x2 > img.shape[1]:
                x2 = img.shape[1]  # bottom right ke bahar
            if y2 > img.shape[0]:
                y2 = img.shape[0]  # nichese bahar
            if y1 > img.shape[0]:
                y1 = img.shape[0]  # nichese bahar
            if y1 == y2:
                y1 = 0
            temp = 0
            if y1 > y2:
                temp = y1
                y1 = y2
                y2 = temp

            # Re-calculate the width and height of the pant image(to resize the image when it wud be pasted)
            pantWidth = int(abs(x2 - x1))
            pantHeight = int(abs(y2 - y1))
            x1 = int(x1)
            x2 = int(x2)
            y1 = int(y1)
            y2 = int(y2)

            # Re-size the original image and the masks to the pant sizes

            # resize all,the masks you made,the originla image,everything
            pant = cv2.resize(imgpant, (pantWidth, pantHeight),
                              interpolation=cv2.INTER_AREA)
            mask = cv2.resize(orig_mask, (pantWidth, pantHeight),
                              interpolation=cv2.INTER_AREA)
            mask_inv = cv2.resize(
                orig_mask_inv, (pantWidth, pantHeight), interpolation=cv2.INTER_AREA)

```

```python
            # take ROI for pant from background equal to size of pant image
                roi = img[y1:y2, x1:x2]
                # roi_bg contains the original image only where the pant is not
                # in the region that is the size of the pant.
                num = roi
                roi_bg = cv2.bitwise_and(roi, num, mask=mask_inv)
                # roi_fg contains the image of the pant only where the pant is
                roi_fg = cv2.bitwise_and(pant, pant, mask=mask)
                # join the roi_bg and roi_fg
                dst = cv2.add(roi_bg, roi_fg)
                # place the joined image, saved to dst back over the original image

                # Blur the original input  image/feed when filter is superimposed

                # slecting the area to blurr
                top = img[0:y, 0:resizewidth]
                bottom = img[y+h:resizeheight, 0:resizewidth]
                midleft = img[y:y+h, 0:x]
                midright = img[y:y+h, x+w:resizewidth]
                blurvalue = 5

```

```python
            # Apply gaussian blurr
            top = cv2.GaussianBlur(top, (blurvalue, blurvalue), 0)
            bottom = cv2.GaussianBlur(bottom, (blurvalue, blurvalue), 0)
            midright = cv2.GaussianBlur(midright, (blurvalue, blurvalue), 0)
            midleft = cv2.GaussianBlur(midleft, (blurvalue, blurvalue), 0)

            img[0:y, 0:resizewidth] = top
            img[y+h:resizeheight, 0:resizewidth] = bottom
            img[y:y+h, 0:x] = midleft
            img[y:y+h, x+w:resizewidth] = midright
            img[y1:y2, x1:x2] = dst

    # |||||||||||||||||||||||||||||||SHIRT||||||||||||||||||||||||||||||||||||||||

            shirtWidth = 3 * w  # approx wrt face width
            # preserving aspect ratio of original image..
            shirtHeight = shirtWidth * origshirtHeight / origshirtWidth
            # Center the shirt..just random calculations..
            x1s = x-w
            x2s = x1s+3*w
            y1s = y+h
            y2s = y1s+h*4
            # Check for clipping(whetehr x1 is coming out to be negative or not..)

            if x1s < 0:
                x1s = 0
            if x2s > img.shape[1]:
                x2s = img.shape[1]
            if y2s > img.shape[0]:
                y2s = img.shape[0]
            temp = 0
            if y1s > y2s:
                temp = y1s
                y1s = y2s
                y2s = temp

            # Re-calculate the width and height of the shirt image(to resize the image when it wud be pasted)
            shirtWidth = int(abs(x2s - x1s))
            shirtHeight = int(abs(y2s - y1s))
            y1s = int(y1s)
            y2s = int(y2s)
            x1s = int(x1s)
            x2s = int(x2s)
```

```python

            # Re-size the original image and the masks to the shirt sizes
            # resize all,the masks you made,the originla image,everything
            shirt = cv2.resize(
                imgshirt, (shirtWidth, shirtHeight), interpolation=cv2.INTER_AREA)
            mask = cv2.resize(
                orig_masks, (shirtWidth, shirtHeight), interpolation=cv2.INTER_AREA)
            masks_inv = cv2.resize(
                orig_masks_inv, (shirtWidth, shirtHeight), interpolation=cv2.INTER_AREA)
            # take ROI for shirt from video equal to size of shirt image
            rois = img[y1s:y2s, x1s:x2s]
            # roi_bg contains the original image only where the shirt is not
            # in the region that is the size of the shirt.
            num = rois
            roi_bgs = cv2.bitwise_and(rois, num, mask=masks_inv)
            # roi_fg contains the image of the shirt only where the shirt is
            roi_fgs = cv2.bitwise_and(shirt, shirt, mask=mask)
            # join the roi_bg and roi_fg
            dsts = cv2.add(roi_bgs, roi_fgs)
            # place the joined image, saved to dst back over the original image
            img[y1s:y2s, x1s:x2s] = dsts
            # print "blurring"

            break
        cv2.putText(img, "Shirt Width: {}".format(shirtWidth), (50, 50),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2, cv2.LINE_4)
        cv2.putText(img, "Shirt Height: {}".format(shirtHeight), (50, 80),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2, cv2.LINE_4)
        cv2.putText(img, "Pant Width: {}".format(pantWidth, pantHeight),
                    (50, 110), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2, cv2.LINE_4)
        cv2.putText(img, "Pant Height: {}".format(pantHeight), (50, 140),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2, cv2.LINE_4)

        cv2.imshow("img", img)
        if cv2.waitKey(100) == ord('q'):
            break

    cap.release()                           # Destroys the cap object
    cv2.destroyAllWindows()                 # Destroys all the windows created by imshow

    return render_template('index.html', size_detected=random.choice(['L', 'XL']))


if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True, port=5000)
```

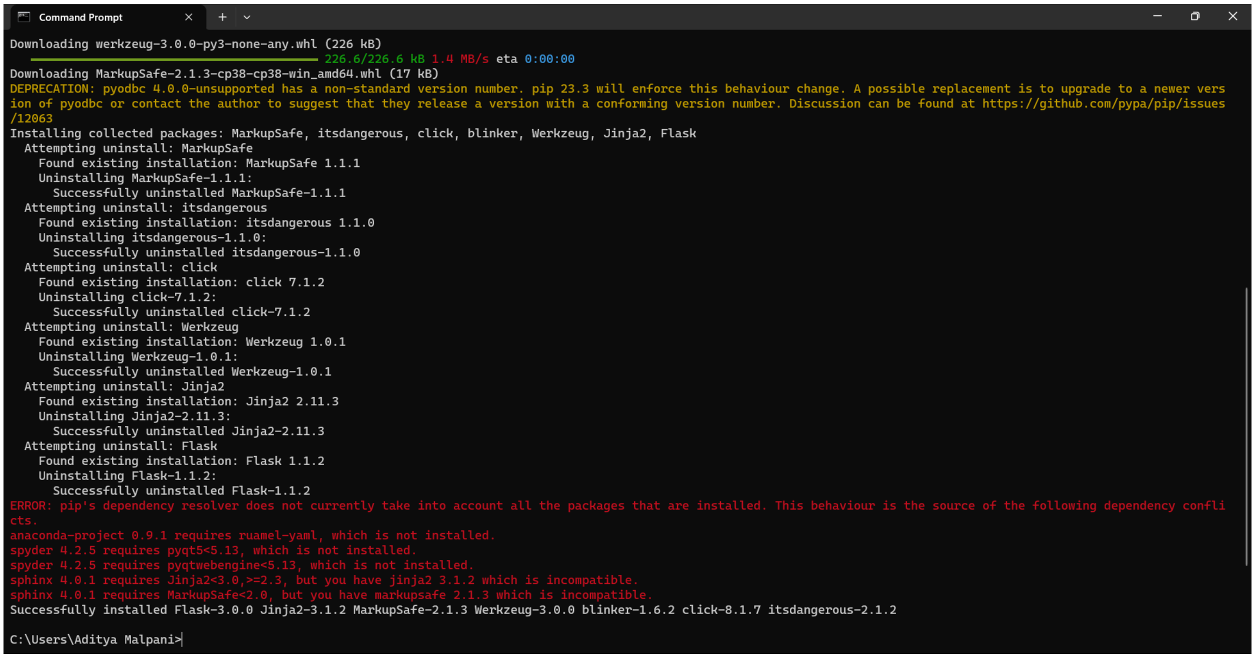
Output and Results:

Installing and checking Python, PIP, OpenCV, and Flask:

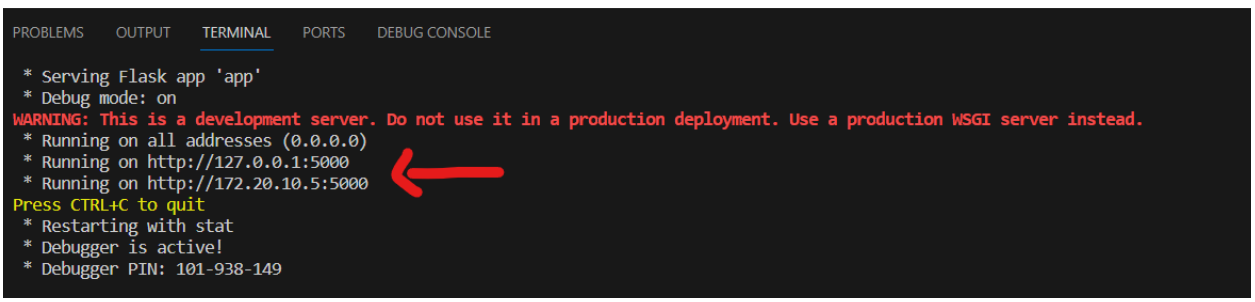
```
Command Prompt - python

Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Aditya Malpani>python --version
Python 3.8.8

C:\Users\Aditya Malpani>python --V
unknown option --V
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Try `python -h' for more information.

C:\Users\Aditya Malpani>python get-pip.py
python: can't open file 'get-pip.py': [Errno 2] No such file or directory

C:\Users\Aditya Malpani>curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 2544k  100 2544k    0     0  1301k      0  0:00:01  0:00:01 --:--:-- 1302k

C:\Users\Aditya Malpani>python get-pip.py
Collecting pip
  Obtaining dependency information for pip from https://files.pythonhosted.org/packages/50/c2/e06851e8cc28dcad7c155f4753da8833ac06a5c704c109313b8d5a62968a/p
ip-23.2.1-py3-none-any.whl.metadata
  Downloading pip-23.2.1-py3-none-any.whl.metadata (4.2 kB)
Downloading pip-23.2.1-py3-none-any.whl (2.1 MB)
   ---------------------------------------- 2.1/2.1 MB 604.0 kB/s eta 0:00:00
DEPRECATION: pyodbc 4.0.0-unsupported has a non-standard version number. pip 23.3 will enforce this behaviour change. A possible replacement is to upgrade t
o a newer version of pyodbc or contact the author to suggest that they release a version with a conforming version number. Discussion can be found at https:
//github.com/pypa/pip/issues/12063
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 21.0.1
    Uninstalling pip-21.0.1:

pip 23.2.1 from C:\ProgramData\Anaconda3\lib\site-packages\pip (python 3.8)
```

```
pip 23.2.1 from C:\ProgramData\Anaconda3\lib\site-packages\pip (python 3.8)

C:\Users\Aditya Malpani>pip install opencv-python
Collecting opencv-python
  Obtaining dependency information for opencv-python from https://files.pythonhosted.org/packages/38/d2/3e8c13ffc37ca5ebc6f382b242b44acb43eb489042e1728407ac
3904e72f/opencv_python-4.8.1.78-cp37-abi3-win_amd64.whl.metadata
3904e72f/opencv_python-4.8.1.78-cp37-abi3-win_amd64.whl.metadata
Requirement already satisfied: numpy>=1.17.0 in c:\programdata\anaconda3\lib\site-packages (from opencv-python) (1.20.1)Downloading opencv_python-4.8.1.78-c
p37-abi3-win_amd64.whl (38.1 MB)
   ---------------------------------------- 38.1/38.1 MB 3.2 MB/s eta 0:00:00
DEPRECATION: pyodbc 4.0.0-unsupported has a non-standard version number. pip 23.3 will enforce this behaviour change. A possible replacement is to upgrade t
o a newer version of pyodbc or contact the author to suggest that they release a version with a conforming version number. Discussion can be found at https:
//github.com/pypa/pip/issues/12063
//github.com/pypa/pip/issues/12063
Successfully installed opencv-python-4.8.1.78

C:\Users\Aditya Malpani>python
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated.  Libraries may fail to load.  To activate this environment
please see https://conda.io/activation
please see https://conda.io/activation
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> print(cv2.__version__)
4.8.1
>>>
```

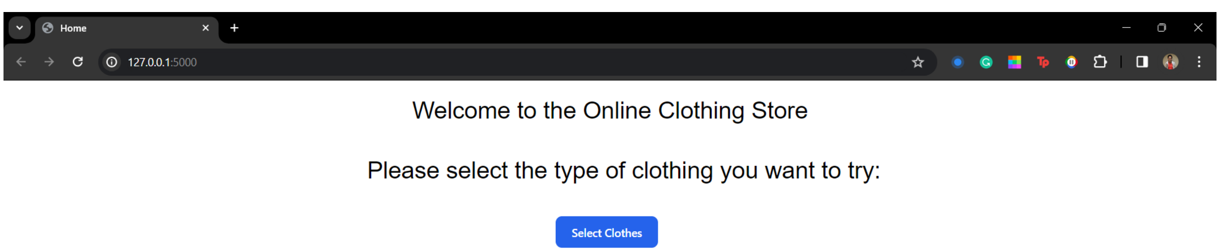
```
Command Prompt

Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Aditya Malpani>pip install -U Flask
Requirement already satisfied: Flask in c:\programdata\anaconda3\lib\site-packages (1.1.2)
Collecting Flask
  Obtaining dependency information for Flask from https://files.pythonhosted.org/packages/36/42/015c23096649b908c809c69388a805a571a3bea44362fe87e33fc3afa01f
/flask-3.0.0-py3-none-any.whl.metadata
  Downloading flask-3.0.0-py3-none-any.whl.metadata (3.6 kB)
Collecting Werkzeug>=3.0.0 (from Flask)
  Obtaining dependency information for Werkzeug>=3.0.0 from https://files.pythonhosted.org/packages/b6/a5/54b01f663d60d5334f6c9c87c26274e94617a4fd463d812463
626423b10d/werkzeug-3.0.0-py3-none-any.whl.metadata
  Downloading werkzeug-3.0.0-py3-none-any.whl.metadata (4.1 kB)
Collecting Jinja2>=3.1.2 (from Flask)
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
     ---------------------------------------- 133.1/133.1 kB 1.1 MB/s eta 0:00:00
Collecting itsdangerous>=2.1.2 (from Flask)
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting click>=8.1.3 (from Flask)
  Obtaining dependency information for click>=8.1.3 from https://files.pythonhosted.org/packages/00/2e/d53fa4befbf2cfa713304affc7ca780ce4fc1fd8710527771b583
11a3229/click-8.1.7-py3-none-any.whl.metadata
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from Flask)
  Downloading blinker-1.6.2-py3-none-any.whl (13 kB)
Requirement already satisfied: importlib-metadata>=3.6.0 in c:\programdata\anaconda3\lib\site-packages (from Flask) (3.10.0)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-packages (from click>=8.1.3->Flask) (0.4.4)Requirement already satisfied: zipp>
=0.5 in c:\programdata\anaconda3\lib\site-packages (from importlib-metadata>=3.6.0->Flask) (3.4.1)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->Flask)
  Obtaining dependency information for MarkupSafe>=2.0 from https://files.pythonhosted.org/packages/74/a3/54fc60ee2da3ab6d68b1b2daf4897297c597840212ee126e68
a4eb89fcd7/MarkupSafe-2.1.3-cp38-cp38-win_amd64.whl.metadata
  Downloading MarkupSafe-2.1.3-cp38-cp38-win_amd64.whl.metadata (3.1 kB)
Downloading flask-3.0.0-py3-none-any.whl (99 kB)
   ---------------------------------------- 99.7/99.7 kB 2.9 MB/s eta 0:00:00
Downloading click-8.1.7-py3-none-any.whl (97 kB)
   ---------------------------------------- 97.9/97.9 kB 5.5 MB/s eta 0:00:00
Downloading werkzeug-3.0.0-py3-none-any.whl (226 kB)
   ---------------------------------------- 226.6/226.6 kB 1.4 MB/s eta 0:00:00
```



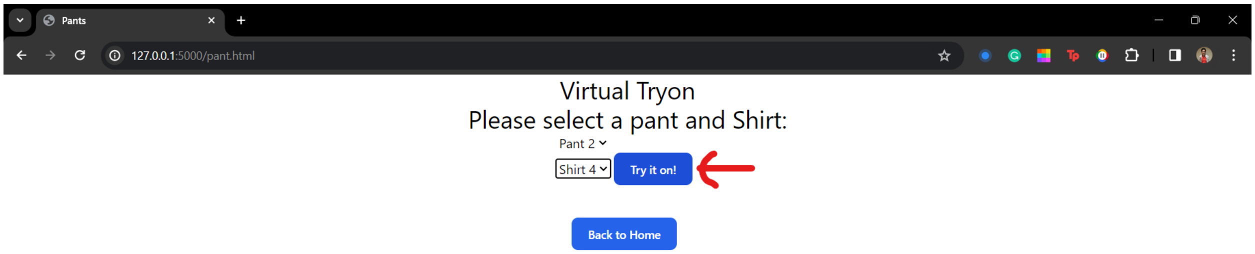
Implementing Code:



After running the code in Visual Studio Code, it generates a website link. Ctrl + Click on the link to redirect to the website.



Landing page of the Redirected Website



After selecting your desired clothes, click on TRY IT NOW!

A window pops up after you grant camera access where our prototype runs successfully! Happy Trying-On!

Limitations and scope for future work:

The research on virtual try-on and dress fitness prediction using DL/ML techniques has some limitations that warrant consideration in future work. First, the accuracy of the predictions heavily relies on the quality and representativeness of the training data. If the training data is biased, incomplete, or limited in diversity, it may affect the performance and generalization of the DL/ML models. Additionally, the reliance on digital avatars or virtual representations of users may not fully capture the individual variations in body shape, size, and posture, which could impact the accuracy of fitness prediction.

Another limitation is the computational complexity and resource requirements of DL/ML models. Deep learning models often require substantial computing power, memory, and storage, which may not be readily available to all users or in all environments. This can limit the scalability and accessibility of virtual try-on and dress fitness prediction systems, particularly for smaller retailers or users with limited computing resources.

Furthermore, virtual try-on technology is heavily dependent on the availability and quality of 3D models of clothing items. Creating accurate and realistic 3D models of various types of clothing, fabrics, and styles can be challenging and time-consuming, and may require specialized expertise or equipment. The lack of comprehensive and high-quality 3D models could impact the visual fidelity and realism of the virtual try-on experience.

Despite these limitations, there are several promising areas for future work in this field. This includes the development of more diverse and comprehensive datasets for training and evaluation, improving clothing modeling algorithms to enhance accuracy and realism, enhancing the user experience through advancements in virtual try-on technologies, and addressing ethical and social concerns. Further research can also explore novel DL/ML techniques, such as GANs and reinforcement learning, for virtual try-on and dress fitness prediction. Additionally, investigating the potential of incorporating user feedback, customization, and personalization in virtual try-on systems can enhance their practical applications in the fashion industry and beyond. Overall, addressing the limitations and exploring new avenues of research can further advance the field of virtual try-on and dress fitness prediction using DL/ML techniques, making them more reliable, user-friendly, and ethically responsible.

Conclusion:

Virtual try-on and dress fitness prediction using DL/ML techniques hold great promise in revolutionizing the fashion industry by providing virtual fitting experiences for online shoppers. These technologies have shown significant advancements in recent years, enabling users to virtually try on clothes and predict their fitness with accuracy. However, there are still limitations that need to be addressed, such as data availability, clothing modeling complexities, technology limitations, user experience, and real-world deployment challenges.

Despite these limitations, the potential for future research and advancements in DL/ML techniques, technologies, and user-centric design is vast. Further development of large and diverse datasets, incorporation of physics-based simulations, multi-modal inputs, improved garment modeling techniques, optimization of user experience and interaction design, and practical deployment and integration in real-world retail settings can enhance the effectiveness and adoption of virtual try-on and dress fitness prediction technologies.

With continued research and innovation, virtual try-on and dress fitness prediction using DL/ML have the potential to transform the online shopping experience, reduce returns, and improve customer satisfaction in the fashion industry. As technology continues to evolve, these technologies are expected to play an increasingly significant role in the future of fashion retail, offering personalized and immersive virtual fitting experiences for shoppers, and driving the growth and competitiveness of the fashion industry in the digital era.