Appendix A: Sample Code

Flasktry.py

#import libraries for this project#

```
from flask import Flask, render_template, request
import json
from flask_cors import CORS
import numpy as np
import cv2
from math import floor
app = Flask(__name__)
CORS(app)
```

#Call the HTML File for UI & get User inputs#

```
@app.route('/')
def index():
return render_template('index.html')
@app.route('/shirt.html')
def plot():
return render_template('shirt.html')
@app.route('/pant.html')
def ploty():
return render_template('pant.html')
@app.route('/predict', methods=['GET','POST'])
```

#Process the Prediction by using user data#

```
def predict():
shirtno = int(request.form["shirt"])
pantno = int(request.form["pant"])
cv2.cv2.waitKey(1)
cap=cv2.cv2.VideoCapture(0)
ih=shirtno
i=pantno
```

#Perform the user selected Selection(Shirt color & Pant color)#

```
while True:
imgarr=["shirt1.png",'shirt2.png','shirt51.jpg','shirt6']
imgshirt = cv2.cv2.imread(imgarr[ih-1],1)
if ih==3:
shirtgray = cv2.cv2.cvtColor(imgshirt,cv2.cv2.COLOR_BGR2GRAY)
ret, orig_masks_inv = cv2.cv2.threshold(shirtgray,200 , 255,
cv2.cv2.THRESH_BINARY)
orig_masks = cv2.cv2.bitwise_not(orig_masks_inv)
else:
shirtgray = cv2.cv2.cvtColor(imgshirt,cv2.cv2.COLOR_BGR2GRAY)
ret, orig_masks = cv2.cv2.threshold(shirtgray,0 ,
255,cv2.cv2.THRESH_BINARY)
orig_masks_inv = cv2.cv2.bitwise_not(orig_masks)
origshirtHeight, origshirtWidth = imgshirt.shape[:2]
imgarr=["pant7.jpg",'pant21.png']
imgpant = cv2.cv2.imread(imgarr[i-1],1)
imgpant=imgpant[:,:,0:3]
pantgray = cv2.cv2.cvtColor(imgpant,cv2.cv2.COLOR_BGR2GRAY)
if i==1:
```

```
ret, orig_mask = cv2.cv2.threshold(pantgray,100 , 255,
cv2.cv2.THRESH_BINARY)
orig_mask_inv = cv2.cv2.bitwise_not(orig_mask)
else:
ret, orig_mask = cv2.cv2.threshold(pantgray,50 , 255,
cv2.cv2.THRESH_BINARY)
orig_mask_inv = cv2.cv2.bitwise_not(orig_mask)
origpantHeight, origpantWidth = imgpant.shape[:2]
face_cascade=cv2.cv2.CascadeClassifier('haarcascade_frontalface_
default.xml')
ret,img=cap.read()
height = img.shape[0]
width = img.shape[1]
```
#Resize(adjusting) Clothes for perfect fitting to user(height, width)#
```
resizewidth = int(width*3/2)
resizeheight = int(height*3/2)
cv2.cv2.namedWindow("img",cv2.cv2.WINDOW_NORMAL)
cv2.cv2.resizeWindow("img", (int(width*3/2), int(height*3/2)))
gray=cv2.cv2.cvtColor(img,cv2.cv2.COLOR_BGR2GRAY)
faces=face_cascade.detectMultiScale(gray,1.3,5)
for (x,y,w,h) in faces:
cv2.cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
cv2.cv2.rectangle(img,(100,200),(312,559),(255,255,255),2)
pantWidth = 3 * w
pantHeight = pantWidth * origpantHeight / origpantWidth
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
if x1 < 0:  x1 = 0
if x2 > img.shape[1]:  x2 =img.shape[1]
if y2 > img.shape[0]:  y2 =img.shape[0]
if y1 > img.shape[0]:  y1 =img.shape[0]
if y1==y2:  y1=0   temp=0
if y1>y2:   temp=y1 y1=y2   y2=temp
pantWidth = int(abs(x2 - x1))
pantHeight = int(abs(y2 - y1))
x1 = int(x1)   x2 = int(x2)
y1 = int(y1)   y2 = int(y2)
pant = cv2.cv2.resize(imgpant, (pantWidth,pantHeight),
interpolation = cv2.cv2.INTER_AREA)
mask = cv2.cv2.resize(orig_mask, (pantWidth,pantHeight),
interpolation = cv2.cv2.INTER_AREA)
mask_inv = cv2.cv2.resize(orig_mask_inv, (pantWidth,pantHeight),
interpolation = cv2.cv2.INTER_AREA)
```
#Performing Background Blur for Virtual Trial period#
```
roi = img[y1:y2, x1:x2]
num=roi
roi_bg = cv2.cv2.bitwise_and(roi,num,mask = mask_inv)
roi_fg = cv2.cv2.bitwise_and(pant,pant,mask = mask)
dst = cv2.cv2.add(roi_bg,roi_fg)
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
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)
```
#Re-sizing your(user's) selection until perfect#
```
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
shirtWidth = 3 * w
shirtHeight = shirtWidth * origshirtHeight / origshirtWidth
x1s = x-w  x2s =x1s+3*w   y1s = y+h  y2s = y1s+h*4
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
shirtWidth = int(abs(x2s - x1s))
shirtHeight = int(abs(y2s - y1s))
y1s = int(y1s)  y2s = int(y2s)
x1s = int(x1s)  x2s = int(x2s)
shirt = cv2.cv2.resize(imgshirt, (shirtWidth,shirtHeight),
interpolation = cv2.cv2.INTER_AREA)
mask = cv2.cv2.resize(orig_masks, (shirtWidth,shirtHeight),
interpolation = cv2.cv2.INTER_AREA)
masks_inv = cv2.cv2.resize(orig_masks_inv,(shirtWidth,
shirtHeight),interpolation = cv2.cv2.INTER_AREA)
rois = img[y1s:y2s, x1s:x2s]
num=rois
roi_bgs = cv2.cv2.bitwise_and(rois,num,mask = masks_inv)
roi_fgs = cv2.cv2.bitwise_and(shirt,shirt,mask = mask)
dsts = cv2.cv2.add(roi_bgs,roi_fgs)
img[y1s:y2s, x1s:x2s] = dsts  break
cv2.cv2.imshow("img",img)
```
#Close the Trial Show#
```
if cv2.cv2.waitKey(100) == ord('q'): break
cap.release()
cv2.cv2.destroyAllWindows()
return render_template('index.html')
if __name__ ==
'__main__':app.run(host='0.0.0.0',debug=True,port=5000)
```