Name- Devansh Sharma
Roll. no - 2215500058
University - GLA University
Mail- devansh.sharma_cs.aiml22@gla.ac.in

Task - 1 - Remove Background.

```
pip install rembg
pip install onnxruntime
```

`Rembeg` **-** A Python tool to remove image backgrounds automatically.Uses deep learning models (like U²-Net) for background removal.After installation, you can use it via **CLI** or **Python code**.

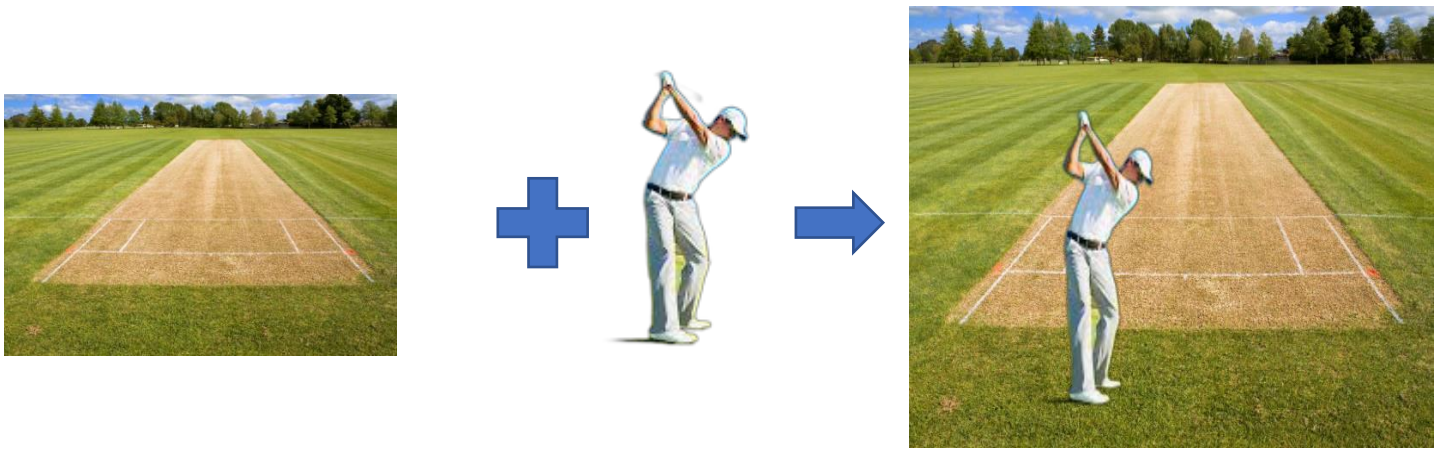
`Onnxruntime` **-** A runtime for executing ONNX (Open Neural Network Exchange) models.`rembg` depends on `onnxruntime` for running its neural network models efficiently.

```
from rembg import remove

from PIL import Image

input_path = "person.jpeg"

output_path = "person_no_bg.png"

input_image = Image.open(input_path)

output_image = remove(input_image)

output_image.save(output_path)
```

In this step we are removing background from the image and identifying the person.

Task – 2 – Choose a scenery and add this image to scenery

```
import cv2
import numpy as np
background = cv2.imread("scenery.jpg")
person = cv2.imread("person_no_bg.png", cv2.IMREAD_UNCHANGED)
bh, bw = background.shape[:2]
scale_percent = int(input("Enter the % of background height the person should cover: "))
target_height = int(bh * (scale_percent / 100.0))
scale_factor = target_height / person.shape[0]
person = cv2.resize(person, (int(person.shape[1] * scale_factor), target_height))
ph, pw = person.shape[:2]
x_offset = (bw - pw) // 2
y_offset = bh - ph - 10
if person.shape[2] == 4:
    alpha = person[:, :, 3] / 255.0
    alpha_inv = 1 - alpha
    person_rgb = person[:, :, :3]
else:
    alpha = np.ones((ph, pw))
    alpha_inv = 1 - alpha
    person_rgb = person
roi = background[y_offset:y_offset+ph, x_offset:x_offset+pw]
for c in range(3):
    roi[:, :, c] = alpha * person_rgb[:, :, c] + alpha_inv * roi[:, :, c]
background[y_offset:y_offset+ph, x_offset:x_offset+pw] = roi
cv2.imwrite("final_output.png", background)

print("Saved final_output.png").
```



Task - 3 - Add shadow to the final output.

```
!pip install --quiet mediapipe opencv-python-headless pillow numpy
```

mediapipe- A Google framework for building computer vision pipelines (face detection, hand tracking, pose estimation, etc.).

opencv - python-headless- OpenCV library for image processing and computer vision, but without GUI dependencies (useful in servers and Colab).

pillow (PIL) - A Python imaging library for opening, editing, and saving image files.

numpy - A core library for numerical computations in Python (used for handling arrays and matrices).

```
import cv2
import mediapipe as mp
import numpy as np
from PIL import Image, ImageFilter, ImageEnhance
import os
def add_shadow_to_person(
    image_path,
    output_path="image_with_shadow.png",
```

```
angle_deg=30
shadow_distance=40,
shadow_opacity=100,
blur_radius=15
img_bgr = cv2.imread(image_path)
if img_bgr is None:
    raise FileNotFoundError(f"Could not load image: {image_path}")
h, w = img_bgr.shape[:2]
mp_selfie = mp.solutions.selfie_segmentation.SelfieSegmentation(model_selection=1)
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
result = mp_selfie.process(img_rgb)
person_mask = (result.segmentation_mask > 0.5).astype(np.uint8) * 255
img_rgba = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2BGRA)
img_rgba[:, :, 3] = person_mask  # alpha channel = person mask
person_pil = Image.fromarray(cv2.cvtColor(img_rgba, cv2.COLOR_BGRA2RGBA))
alpha = person_pil.split()[3]
shadow = Image.new("RGBA", person_pil.size, (0, 0, 0, 0))
shadow.paste((0, 0, 0, shadow_opacity), mask=alpha)
shadow = shadow.filter(ImageFilter.GaussianBlur(blur_radius))
dx = int(np.cos(np.radians(angle_deg)) * shadow_distance)
dy = int(np.sin(np.radians(angle_deg)) * shadow_distance)
shadow = shadow.transform(
    person_pil.size, Image.AFFINE, (1, 0, dx, 0, 1, dy), resample=Image.BICUBIC
)
base_img = Image.open(image_path).convert("RGBA")
final_img = Image.alpha_composite(base_img, shadow)
final_img = Image.alpha_composite(final_img, person_pil)
final_img.save(output_path)
print(f"Saved: {output_path}")
```

return output_path

This is the code for shadow generation now we need to call a function.

```
add_shadow_to_person(
    input_image,
    output_image,
    angle_deg=200,      # 200° = left-down direction
    shadow_distance=35,  # distance of shadow
    shadow_opacity=150,  # darkness
    blur_radius=1       # softness
)
```

SHADOW