LAMPIRAN

1. Script app.py

```
from flask import Flask, render_template, Response, request, jsonify, send_file
from flask_socketio import SocketIO
import cv2
import json
import mediapipe as mp
import numpy as np
from scipy.spatial import procrustes
import base64
import os
app = Flask(__name__)
socketio = SocketIO(app)
mp_pose = mp.solutions.pose
mp_drawing = mp.solutions.drawing_utils
image_name = "Camel"
image = None
# Extract landmarks from JSON data
landmarks_from_json = []
the_landmarks = None
dataset = {"name": "", "ket": ""}
similarity = 0
all_data = []
def load_image_and_landmarks(image_name):
    global image, landmarks_from_json, the_landmarks, all_data
    landmarks_from_json = []  # Clear previous landmarks
```

```
    # Load JSON data
    with open('data_yoga.json') as f:
        data = json.load(f)
        all_data = data
    # Load the image and landmarks
    for the_data in data:
        if the_data['name'] == image_name:
            for lm in the_data['landmarks']:
                landmarks_from_json.append([lm['coordinates'][0], lm['coordinates'][1]])
            the_landmarks = the_data['landmarks']
            image = cv2.imread(the_data['image_name'])
            dataset["name"] = the_data['name']
            dataset["ket"] = the_data['ket']
# Define the function to calculate the color based on similarity
def calculate_color(similarity):
    if similarity < 80:
        return (0, 0, 255)  # Red
    else:
        normalized_similarity = (similarity - 55) / 45
        red = int((1 - normalized_similarity) * 255)
        green = int(normalized_similarity * 255)
        return (0, green, red)
def resample_landmarks(landmarks, target_length):
    idxs = np.linspace(0, len(landmarks) - 1, target_length).astype(int)
    return [landmarks[i] for i in idxs]
```

```
def calculate_similarity(landmarks1, landmarks2):
    if not landmarks1 or not landmarks2:
        return 0
    len1 = len(landmarks1)
    len2 = len(landmarks2)
    # Resample landmarks to ensure they have the same number of points
    if len1 != len2:
        if len1 < len2:
            landmarks2 = resample_landmarks(landmarks2, len1)
        else:
            landmarks1 = resample_landmarks(landmarks1, len2)
    # Convert to numpy arrays
    landmarks1 = np.array(landmarks1)
    landmarks2 = np.array(landmarks2)
    # Normalize landmarks by removing the mean
    norm_landmarks1 = landmarks1 - np.mean(landmarks1, axis=0)
    norm_landmarks2 = landmarks2 - np.mean(landmarks2, axis=0)
    # Perform Procrustes analysis to align the shapes
    _, mtx1, mtx2 = procrustes(norm_landmarks1, norm_landmarks2)
    # Calculate the Euclidean distances between corresponding landmarks
    dists = np.linalg.norm(mtx1 - mtx2, axis=1)
    # Calculate the similarity as the inverse of the average distance
    avg_dist = np.mean(dists)
    similarity = max(0, 1 - avg_dist)
    # Scale the similarity to a percentage
```

```
    similarity_percentage = similarity * 100
    return similarity_percentage
def draw_landmarks(image, landmarks):
    global similarity
    annotated_image = image.copy()
    for landmark in landmarks:
        landmark_x = int(landmark['coordinates'][0] * annotated_image.shape[1])
        landmark_y = int(landmark['coordinates'][1] * annotated_image.shape[0])
        cv2.circle(annotated_image, (landmark_x, landmark_y), 5, (0, 255, 0), -1)
    connections = mp_pose.POSE_CONNECTIONS
    for connection in connections:
        start_idx = connection[0]
        end_idx = connection[1]
        if 0 <= start_idx < len(landmarks) and 0 <= end_idx < len(landmarks):
            start_landmark = landmarks[start_idx]['coordinates']
            end_landmark = landmarks[end_idx]['coordinates']
            start_x = int(start_landmark[0] * annotated_image.shape[1])
            start_y = int(start_landmark[1] * annotated_image.shape[0])
            end_x = int(end_landmark[0] * annotated_image.shape[1])
            end_y = int(end_landmark[1] * annotated_image.shape[0])
            cv2.line(annotated_image, (start_x, start_y), (end_x, end_y), (0, 255, 0), 2)
    return annotated_image
def generate_frames():
    global similarity
    cap = cv2.VideoCapture(1)
```

```
    with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as
pose:
        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break
            image_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image_rgb.flags.writeable = False
            results = pose.process(image_rgb)
            image_rgb.flags.writeable = True
            image_bgr = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2BGR)
            the_color = calculate_color(similarity)
            if results.pose_landmarks:
                mp_drawing.draw_landmarks(
                    image_bgr,
                    results.pose_landmarks,
                    mp_pose.POSE_CONNECTIONS,
                    mp_drawing.DrawingSpec(color=(0, 255, 0), thickness=4, circle_radius=2),
                    mp_drawing.DrawingSpec(color=(the_color), thickness=4, circle_radius=2),
                )
                landmarks_from_webcam = []
                for lm in results.pose_landmarks.landmark:
                    landmarks_from_webcam.append([lm.x, lm.y])
                                    similarity = calculate_similarity(landmarks_from_json,
landmarks_from_webcam)
```

```
                    cv2.putText(image_bgr, f'Similarity: {similarity:.2f}%', (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
            ret, buffer = cv2.imencode('.jpg', image_bgr)
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
    cap.release()
@app.route('/')
def index():
    image_name = request.args.get('image_name', 'Camel')
    previous = None
    next = None
    load_image_and_landmarks(image_name)
    current_index = 0
    for index, data in enumerate(all_data):
        if data['name'] == image_name:
            current_index = index
            break
    if current_index == 0:
        previous = all_data[-1]['name']
    else:
        previous = all_data[current_index - 1]['name']

    if current_index == len(all_data) - 1:
        next = all_data[0]['name']
```

```
    else:
        next = all_data[current_index + 1]['name']
    annotated_image = draw_landmarks(image, the_landmarks)
    _, buffer = cv2.imencode('.jpg', annotated_image)
    img_str = base64.b64encode(buffer).decode('utf-8')
    return render_template('index2.html', img_str=img_str, previous=previous, next=next)
@app.route('/video_feed')
def video_feed():
        return Response(generate_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')
@app.route('/getdata', methods=['GET'])
def getdata():
    return jsonify(all_data)
@app.route('/similarity', methods=['GET'])
def get_similarity():
    global similarity
    return {'similarity': similarity, 'data': dataset}
@app.route('/pose_dataset')
def pose_dataset():
    load_image_and_landmarks("Camel")
    return render_template('pose_dataset.html', data=all_data)
@app.route('/show_image')
def show_image():
    image_path = request.args.get('image_path')
    if image_path and os.path.exists(image_path):
```

```
        return send_file(image_path, mimetype='image/jpeg')
    else:
        return "Image not found", 404
if __name__ == '__main__':
    socketio.run(app, debug=True)
```

2. Script Index

```
<!DOCTYPE html>
<html lang="en">
<!-- Mirrored from demo.ninjateam.org/html/my-admin/light/page-starter.html by HTTrack Website Copier/3.x [XR&CO'2014], Wed, 03 Jan 2018 03:48:56 GMT -->
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no">
    <meta name="description" content="">
    <meta name="author" content="">
    <title>Yoga Pose Recognition</title>
    <!-- Main Styles -->
    <link rel="stylesheet" href="static/styles/style.min.css">
    <!-- Material Design Icon -->
    <link rel="stylesheet" href="static/fonts/material-design/css/materialdesignicons.css">

    <!-- mCustomScrollbar -->
```

```
                                    <link rel="stylesheet"
href="static/plugin/mCustomScrollbar/jquery.mCustomScrollbar.min.css">
    <!-- Waves Effect -->
    <link rel="stylesheet" href="static/plugin/waves/waves.min.css">
    <!-- Sweet Alert -->
    <link rel="stylesheet" href="static/plugin/sweet-alert/sweetalert.css">
    <style>
        table {
            border: 1px;
            font-family: arial, sans-serif;
            border-collapse: collapse;
            width: 86%;
            margin: auto;
        }
        td,
        th {
            border: 1px solid black !important;
            padding: 5px;
        }
        tr:nth-child(even) {
            background-color: #dddddd;
        }
    </style>
</head>
<body>
```

```
<div class="main-menu">
    <header class="header">
            <a href="" class="logo"><i class="ico mdi mdi-account-box"></i>Body Tracking</a>
                    <button type="button" class="button-close fa fa-times js__menu_close"></button>
        <div class="user">
            <!-- <a href="#" class="avatar"><img src="static/images/avatar-sm-5.jpg" alt=""><span
                class="status online"></span></a> -->
            <h5 class="name"><a href="">Pengguna</a></h5>
            <h5 class="position">User</h5>
            <!-- /.name -->
            <!-- /.control-wrap -->
        </div>
        <!-- /.user -->
    </header>
    <!-- /.header -->
    <div class="content">
        <div class="navigation">
            <h5 class="title">Navigation</h5>
            <!-- /.title -->
            <ul class="menu js__accordion">
                <li class="current">
```

```
                    <a class="waves-effect ac" href="{{ url_for('index') }}"><i class="menu-
icon mdi mdi-account-box"></i><span>Deteksi Pose</span></a>
                </li>
                <li>
                        <a class="waves-effect ac" href="{{ url_for('pose_dataset') }}"><i
class="menu-icon mdi mdi-account-card-details"></i><span>Pose Dataset</span></a>
                </li>
              </ul>
            </div>
            <!-- /.navigation -->
        </div>
        <!-- /.content -->
    </div>
    <!-- /.main-menu -->
    <div class="fixed-navbar">
        <div class="pull-left">
            <button type="button" class="menu-mobile-button glyphicon glyphicon-menu-
hamburger js__menu_mobile"></button>
                    <h1 class="page-title"><span id="today_date"></span> | <span
id="clock"></span></h1>
            <!-- /.page-title -->
        </div>
        <!-- /.pull-left -->
        <div class="pull-right">
        </div>
```

```
        <!-- /.pull-right -->
    </div>
    <!-- /.fixed-navbar -->
    <!-- /#message-popup -->
    <div id="wrapper">
        <div class="main-content">
            <div class="row small-spacing">
                <div class="col-xs-6 col-md-6">
                    <div class="box-content card">
                        <!-- span italic -->
                                <h4 class="box-title">Pose <span id="pose" style="font-style:
italic;"></span></h4>
                        <div class="card-content">
                            <img src="{{ img_str }}" style="width:100%;
height:400px;">
                        </div>
                    </div>
                </div>
                <div class="col-xs-6 col-md-6">
                    <div class="box-content card">
                        <h4 class="box-title">Pose Tubuh <span id="similarity"></span></h4>
                        <div class="card-content">
                                <img src="{{ url_for('video_feed') }}" style="width:100%;
height:400px;">
                        </div>
```

```
            </div>
          </div>
        </div>
        <div class="row small-spacing">
          <div class="col-xs-2 col-md-2 text-center">
            <!-- url with parameters image_name-->
                    <a href="{{ url_for('index', image_name=previous) }}"><button
type="button" class="btn btn-primary btn-xs" id="previous">Sebelumnya</button></a>
          </div>
          <div class="col-xs-8 col-md-8">
            <div class="box-content card">
              <h4 class="box-title">Keterangan</h4>
              <!-- <br> -->
              <div class="card-content">
                <p id="keterangan" style="font-size: 15px;"></p>
              </div>
            </div>
          </div>
          <div class="col-xs-2 col-md-2 text-center">
            <!-- create a next button  and put it on center-->
              <a href="{{ url_for('index', image_name=next) }}"><button type="button"
id="button_next" class="btn btn-primary btn-xs" id="next">Selanjutnya</button></a>
          </div>
        </div>
        <footer class="footer">
```

```
            <ul class="list-inline">
                <li>2024 © Muhammad Irsan.</li>
                <!-- <li><a href="#">Website</a></li>
                <li><a href="#">Facebook</a></li> -->
                <!-- <li><a href="#">Help</a></li> -->
            </ul>
        </footer>
    </div>
    <!-- /.main-content -->
</div><!--/#wrapper -->
 <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
<!--[if lt IE 9]>
    <script src="static/script/html5shiv.min.js"></script>
    <script src="static/script/respond.min.js"></script>
<![endif]-->
<!--
================================================ -->
<!-- Placed at the end of the document so the pages load faster -->
<script src="static/scripts/jquery.min.js"></script>
<script src="static/scripts/modernizr.min.js"></script>
<script src="static/plugin/bootstrap/js/bootstrap.min.js"></script>
                                                                    <script src="static/plugin/mCustomScrollbar/jquery.mCustomScrollbar.concat.min.js"></script>
<script src="static/plugin/nprogress/nprogress.js"></script>
```

```
<script src="static/plugin/sweet-alert/sweetalert.min.js"></script>
<script src="static/plugin/waves/waves.min.js"></script>
<script src="static/scripts/main.min.js"></script>
<script>
    var clockElement = document.getElementById('clock');
    var date = new Date();
    function clock() {
        clockElement.textContent = new Date().toString().slice(15, 24);
    }
    function today_date() {
        $('#today_date').html(date.toDateString());
    }
    today_date();
    setInterval(clock, 1000);
    var counter = 0;
    // // check data every 2 seconds
    setInterval(function () {
        $.ajax({
            url: 'http://127.0.0.1:5000/similarity',
            type: 'GET',
            success: function (data) {
                console.log(data);
                var similarity = data.similarity;
                if (similarity <85) {
                    counter = 0
```

```
            }else{
                counter = counter + 1
            }
            if(counter == 5){
                // click the button_next id
                $('#button_next').trigger('click');
            }
            // only 2 behind .
            similarity = similarity.toFixed(2);
            $('#similarity').html(similarity+ ' % tingkat kesamaan');
            console.log(data.data.name);
            $("#pose").html(data.data.name);
            $('#keterangan').html(data.data.ket);
            // if(data['ket'] == null){
            //  $('#keterangan').html('Tidak ada pose yang terdeteksi');
            // }else{
            //  $('#keterangan').html(data['ket']);
            // }
        }
    });
}, 2000);
// set interval 70 second and change pose
// setInterval(function () {
//     window.location.href="{{ url_for('index', image_name=next) }}";
// }, 70000);
```

```
    </script>
</body>
<!-- Mirrored from demo.ninjateam.org/html/my-admin/light/page-starter.html by HTTrack Website Copier/3.x [XR&CO'2014], Wed, 03 Jan 2018 03:48:56 GMT -->
</html>
```

3. Script Pose Dataset

```
<!DOCTYPE html>
<html lang="en">
<!-- Mirrored from demo.ninjateam.org/html/my-admin/light/page-starter.html by HTTrack Website Copier/3.x [XR&CO'2014], Wed, 03 Jan 2018 03:48:56 GMT -->
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no">
    <meta name="description" content="">
    <meta name="author" content="">
    <title>Yoga Pose Recognition</title>
    <!-- Main Styles -->
    <link rel="stylesheet" href="static/styles/style.min.css">
    <!-- Material Design Icon -->
    <link rel="stylesheet" href="static/fonts/material-design/css/materialdesignicons.css">
    <!-- mCustomScrollbar -->
                                                <link                                rel="stylesheet"
href="static/plugin/mCustomScrollbar/jquery.mCustomScrollbar.min.css">
```

```
    <!-- Waves Effect -->
    <link rel="stylesheet" href="static/plugin/waves/waves.min.css">
    <!-- Sweet Alert -->
    <link rel="stylesheet" href="static/plugin/sweet-alert/sweetalert.css">
    <style>
        table {
            border: 1px;
            font-family: arial, sans-serif;
            border-collapse: collapse;
            width: 86%;
            margin: auto;
        }
        td,
        th {
            border: 1px solid black !important;
            padding: 5px;
        }
        tr:nth-child(even) {
            background-color: #dddddd;
        }
    </style>
</head>
<body>
    <div class="main-menu">
        <header class="header">
```

```
                <a href="" class="logo"><i class="ico mdi mdi-account-box"></i>Body
Tracking</a>
                            <button type="button" class="button-close fa fa-times
js__menu_close"></button>
            <div class="user">
                    <!-- <a href="#" class="avatar"><img src="static/images/avatar-sm-5.jpg"
alt=""><span
                        class="status online"></span></a> -->
                <h5 class="name"><a href="">Pengguna</a></h5>
                <h5 class="position">User</h5>
                <!-- /.name -->
                <!-- /.control-wrap -->
            </div>
            <!-- /.user -->
        </header>
        <!-- /.header -->
        <div class="content">
            <div class="navigation">
                <h5 class="title">Navigation</h5>
                <!-- /.title -->
                <ul class="menu js__accordion">
                    <li>
                        <a class="waves-effect ac" href="{{ url_for('index') }}"><i
                                class="menu-icon mdi mdi-account-box"></i><span>Deteksi
Pose</span></a>
```

```
                </li>
                <li class="current">
                    <a class="waves-effect ac" href="{{ url_for('pose_dataset') }}"><i
                            class="menu-icon mdi mdi-account-card-details"></i><span>Pose
Dataset</span></a>
                </li>
            </ul>
        </div>
        <!-- /.navigation -->
    </div>
    <!-- /.content -->
</div>
<!-- /.main-menu -->
<div class="fixed-navbar">
    <div class="pull-left">
        <button type="button"
                    class="menu-mobile-button glyphicon glyphicon-menu-hamburger
js__menu_mobile"></button>
                <h1 class="page-title"><span id="today_date"></span> | <span
id="clock"></span></h1>
        <!-- /.page-title -->
    </div>
    <!-- /.pull-left -->
    <div class="pull-right">
    </div>
```

```
        <!-- /.pull-right -->
    </div>
    <!-- /.fixed-navbar -->
    <!-- /#message-popup -->
    <div id="wrapper">
        <div class="main-content">
            <div class="row small-spacing">
                <div class="col-xs-2 col-md-2"></div>
                <div class="col-xs-8 col-md-8">
                    <div class="box-content card">
                        <h4 class="box-title">Pose Dataset</h4>
                        <div class="card-content">
                                    <table class="table table-striped table-bordered display"
style="width:100%">
                                <thead>
                                    <tr>
                                        <th>No</th>
                                        <th>Pose</th>
                                        <th>Action</th>
                                    </tr>
                                </thead>
                                <tbody>
                                    {% for pose in data %}
                                    <tr>
                                        <td>{{ loop.index }}</td>
```

```
                        <td>{{ pose['name'] }}</td>
                        <td>
                                <button type="button" class="btn btn-info btn-xs"
onclick="show_pose(`{{ pose['name'] }}`,`{{ pose['image_name'] }}`,`{{ pose['ket']
}}`)">Tampilkan</button>
                        </td>
                      </tr>
                      {% endfor %}
                    </tbody>
                  </table>
                </div>
              </div>
            </div>
            <div class="col-xs-2 col-md-2"></div>
          </div>
          <footer class="footer">
            <ul class="list-inline">
              <li>2024 © Muhammad Irsan.</li>
              <!-- <li><a href="#">Website</a></li>
              <li><a href="#">Facebook</a></li> -->
              <!-- <li><a href="#">Help</a></li> -->
            </ul>
          </footer>
        </div>
        <!-- /.main-content -->
```

```
</div><!--/#wrapper -->
 <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
<!--[if lt IE 9]>
    <script src="static/script/html5shiv.min.js"></script>
    <script src="static/script/respond.min.js"></script>
<![endif]-->
<!--
================================================== -->
<!-- Placed at the end of the document so the pages load faster -->
 <div class="modal fade" id="boostrapModal-1" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
                aria-hidden="true">&times;</span></button>
                 <h4 class="modal-title" id="myModalLabel">Pose <span id="name" style="font-style: italic;"></span></h4>
        </div>
        <div class="modal-body">
          <div class="form-group text-center">
            <img src="" id="image" class="img-responsive">
          </div>
```

```
                <div class="form-group">
                    <p id="ket"></p>
                </div>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default btn-sm waves-effect waves-
light"
                data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>
<script src="static/scripts/jquery.min.js"></script>
<script src="static/scripts/modernizr.min.js"></script>
<script src="static/plugin/bootstrap/js/bootstrap.min.js"></script>
                                                                    <script
src="static/plugin/mCustomScrollbar/jquery.mCustomScrollbar.concat.min.js"></script>
<script src="static/plugin/nprogress/nprogress.js"></script>
<script src="static/plugin/sweet-alert/sweetalert.min.js"></script>
<script src="static/plugin/waves/waves.min.js"></script>
<script src="static/scripts/main.min.js"></script>
<script>
    var clockElement = document.getElementById('clock');
    var date = new Date();
    function clock() {
```

```
            clockElement.textContent = new Date().toString().slice(15, 24);
        }
        function today_date() {
            $('#today_date').html(date.toDateString());
        }
        today_date();
        setInterval(clock, 1000);
        function show_pose(name, image, ket) {
            console.log(name)
            console.log(image)
            console.log(ket)
            $('#image').attr('src', "http://127.0.0.1:5000/show_image?image_path=" + image);
            $('#name').html(name);
            $('#ket').html(ket);
            $('#boostrapModal-1').modal('show')
        }
    </script>
</body>
<!-- Mirrored from demo.ninjateam.org/html/my-admin/light/page-starter.html by HTTrack Website Copier/3.x [XR&CO'2014], Wed, 03 Jan 2018 03:48:56 GMT -->
</html>
```

4. Script Simpan

```
import cv2
import mediapipe as mp
import json
```

```
# Initialize MediaPipe pose module
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5)
mp_drawing = mp.solutions.drawing_utils

# Load and process the image
image_path = 'gerakan/Childs.jpg'
image = cv2.imread(image_path)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image_results = pose.process(image_rgb)

# Extract landmarks from the image
def extract_landmarks(results):
    if not results.pose_landmarks:
        return None
    landmarks = []
    for lm in results.pose_landmarks.landmark:
        landmarks.append((lm.x, lm.y, lm.z))
    return landmarks

image_landmarks = extract_landmarks(image_results)

# Define mapping between landmark indices and body parts
landmark_labels = {
```

0: 'nose',

1: 'left_eye_inner',

2: 'left_eye',

3: 'left_eye_outer',

4: 'right_eye_inner',

5: 'right_eye',

6: 'right_eye_outer',

7: 'left_ear',

8: 'right_ear',

9: 'mouth_left',

10: 'mouth_right',

11: 'left_shoulder',

12: 'right_shoulder',

13: 'left_elbow',

14: 'right_elbow',

15: 'left_wrist',

16: 'right_wrist',

17: 'left_pinky',

18: 'right_pinky',

19: 'left_index',

20: 'right_index',

21: 'left_thumb',

22: 'right_thumb',

23: 'left_hip',

24: 'right_hip',

```
    25: 'left_knee',
    26: 'right_knee',
    27: 'left_ankle',
    28: 'right_ankle',
    29: 'left_heel',
    30: 'right_heel',
    31: 'left_foot_index',
    32: 'right_foot_index'
}

# Map landmark indices to descriptive labels
descriptive_landmarks = [{'body': landmark_labels.get(idx, 'unknown'), 'coordinates':
coord} for idx, coord in enumerate(image_landmarks)]

# Prepare landmark coordinates data
landmark_data = {
    'image_name': image_path,
    'landmarks': descriptive_landmarks
}

# Save landmark coordinates to a variable and print as JSON
landmark_coordinates = json.dumps(landmark_data, indent=4)
print(landmark_coordinates)

# Release resources
```

```
pose.close()
```

5. Script Tampilkan

```
import cv2
import json
import mediapipe as mp
import numpy as np
from scipy.spatial import distance

# Load JSON data
with open('data_yoga.json') as f:
    data = json.load(f)

image_name = "Camel"
image = None

# Extract landmarks from JSON data
landmarks_from_json = []
the_landmarks = None



# Load the image
for the_data in data:
    if the_data['name'] == image_name:
        for lm in the_data['landmarks']:
```

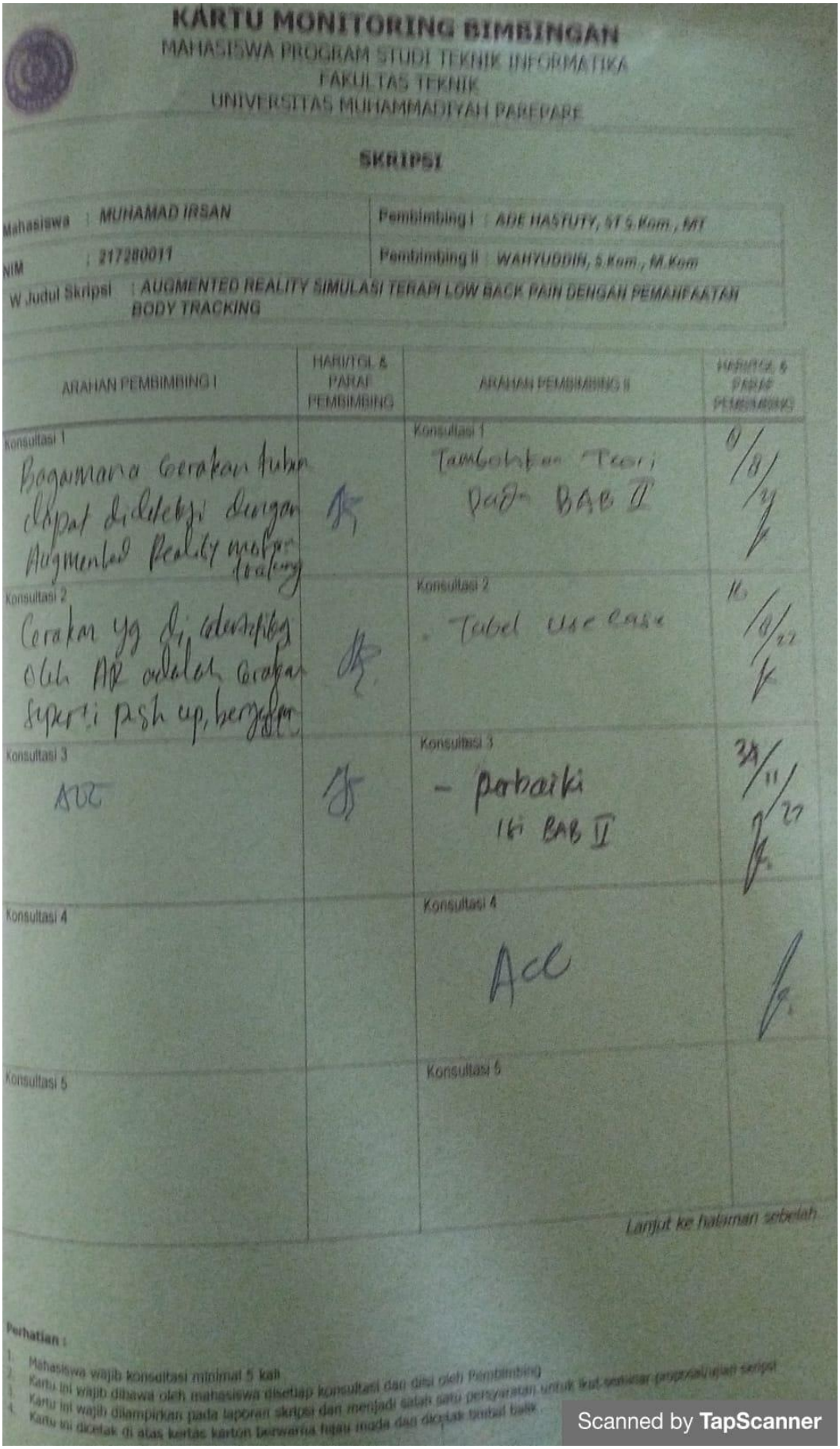
```
            landmarks_from_json.append([lm['coordinates'][0], lm['coordinates'][1]])
        the_landmarks = the_data['landmarks']
        image = cv2.imread(the_data['image_name'])

# Initialize MediaPipe pose module
mp_pose = mp.solutions.pose
mp_drawing = mp.solutions.drawing_utils

# Function to calculate similarity between two sets of landmarks
def calculate_similarity(landmarks1, landmarks2):
    if not landmarks1 or not landmarks2:
        return 0
    # Normalize landmarks
    norm_landmarks1 = np.array(landmarks1) - np.mean(landmarks1, axis=0)
    norm_landmarks2 = np.array(landmarks2) - np.mean(landmarks2, axis=0)
    # Calculate the distance between corresponding landmarks
    dists = [distance.euclidean(lm1, lm2) for lm1, lm2 in zip(norm_landmarks1,
norm_landmarks2)]
    # Calculate similarity as the inverse of the average distance
    similarity = 1 / (1 + np.mean(dists))
    return similarity * 100

# Draw landmarks and connections on the image
def draw_landmarks(image, landmarks):
    annotated_image = image.copy()
```

```
    for landmark in landmarks:
        # Extract landmark coordinates
        landmark_x = int(landmark['coordinates'][0] * annotated_image.shape[1])
        landmark_y = int(landmark['coordinates'][1] * annotated_image.shape[0])
        # Draw a circle at the landmark position
        cv2.circle(annotated_image, (landmark_x, landmark_y), 5, (0, 255, 0), -1)
        # Add text with the body part label
        cv2.putText(annotated_image, landmark['body'], (landmark_x, landmark_y),
cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                    (255, 255, 255), 1)

    # Draw connections between landmarks
    connections = mp_pose.POSE_CONNECTIONS
    for connection in connections:
        start_idx = connection[0]
        end_idx = connection[1]
        if 0 <= start_idx < len(landmarks) and 0 <= end_idx < len(landmarks):
            start_landmark = landmarks[start_idx]['coordinates']
            end_landmark = landmarks[end_idx]['coordinates']
            start_x = int(start_landmark[0] * annotated_image.shape[1])
            start_y = int(start_landmark[1] * annotated_image.shape[0])
            end_x = int(end_landmark[0] * annotated_image.shape[1])
            end_y = int(end_landmark[1] * annotated_image.shape[0])
            cv2.line(annotated_image, (start_x, start_y), (end_x, end_y), (0, 255, 0), 2)
```

```
    return annotated_image



# Annotate and display the image
annotated_image = draw_landmarks(image, the_landmarks)
cv2.imshow('Image with Landmarks and Connections', annotated_image)

# Open webcam
cap = cv2.VideoCapture(0)

with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5)
as pose:
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Convert the BGR image to RGB
        image_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image_rgb.flags.writeable = False

        # Process the image and detect the pose
        results = pose.process(image_rgb)
```

```
        # Convert the RGB image back to BGR
        image_rgb.flags.writeable = True
        image_bgr = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2BGR)

        # Draw the pose annotation on the image
        if results.pose_landmarks:
            mp_drawing.draw_landmarks(
                image_bgr,
                results.pose_landmarks,
                mp_pose.POSE_CONNECTIONS,
                mp_drawing.DrawingSpec(color=(0, 255, 0), thickness=2,
circle_radius=2),
                mp_drawing.DrawingSpec(color=(0, 0, 255), thickness=2,
circle_radius=2),
            )
            # Extract landmarks
            landmarks_from_webcam = []
            for lm in results.pose_landmarks.landmark:
                landmarks_from_webcam.append([lm.x, lm.y])

            # Calculate similarity
            similarity = calculate_similarity(landmarks_from_json,
landmarks_from_webcam)
```

```
            cv2.putText(image_bgr, f'Similarity: {similarity:.2f}%', (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

        # Display the image
        cv2.imshow('Webcam Pose', image_bgr)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
# Release resources
cap.release()
cv2.destroyAllWindows()
cv2.waitKey(1)
```



KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

Mahasiswa : MUHAMAD IRSAN	Pembimbing I : ADE HASTUTY, ST S.Kom., MT
NIM : 217280011	Pembimbing II : WAHYUDDIN, S.Kom., M.Kom
Judul Skripsi : AUGMENTED REALITY SIMULASI TERAPI LOW BACK PAIN DENGAN PEMANFAATAN BODY TRACKING	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 Bagaimana Gerakan tubuh dapat dideteksi dengan Augmented Reality motion tracking		Konsultasi 1 Tambahkan Teori pada BAB II	9/8/22
Konsultasi 2 Gerakan yg di identifikasi oleh AR adalah Gerakan seperti push up, berjalan		Konsultasi 2 - Tabel use case	16/8/22
Konsultasi 3 ACC		Konsultasi 3 - perbaiki lagi BAB II	23/11/22
Konsultasi 4		Konsultasi 4 Acc	
Konsultasi 5		Konsultasi 5	

Lanjut ke halaman sebelah

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak bolak balik

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

Mahasiswa : MUHAMAD IRSAN	Pembimbing I : Ade Hastuty, ST S.Kom., MT
NIM : 217 260 011	Pembimbing II : Wahyuddin, S.Kom., M.Kom
Judul Skripsi : APLIKASI SIMULASI TERAPI LOW BACK PAIN DENGAN PEMANFAATAN BODY TRACKING MENGGUNAKAN OPENCV DAN MEDIAPIPE	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 Abstrak- berisi Rangkuman Isi skripsi		Konsultasi 1 Masukkan halaman deteksi gerakan	
Konsultasi 2 Rumusan masalah terjawab di kesimpulan		Konsultasi 2 Kata Asing Diminimalisir ganti kata pengantar	
Konsultasi 3 Uji White Box dan Black box		Konsultasi 3 ACC	
Konsultasi 4 Saran berisi kekurangan dari aplikasi yg dibuat		Konsultasi 4	
Konsultasi 5 Seminar Hasil 18/07/2024		Konsultasi 5	

Lanjut ke halaman sebelah..

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ji
4. Kartu ini dicetak di atas kertas karton A4 berwarna Hijau muda dan dicetak timbal balik