B. M. S. COLLEGE OF ENGINEERING
Bengaluru - 560019, Karnataka, India.

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"JnanaSangama" Belgaum 590 018, Karnataka, India

A Report on Project-3

Titled

"Yoga Pose Corrector using Computer Vision"

Submitted in the partial fulfilment of academic requirement for the award of

Bachelor of Engineering in Medical Electronics Engineering

Submitted By:

Student Name	USN
Lakshmi H Siddaveer	1BM21MD019
Seggu Harshavardhan	1BM21MD032
Varuni Gorkalkar	1BM21MD042

Under the Guidance of:

Internal Guide:	**External Guide:**
Dr. Beena Ullala Mata B N Professor Department of Medical Electronics Engineering, B M S College of Engineering, Bangalore – 560019	**Sharadha Shetty MS(CS)** Founder Sampoorna Yoga

B. M. S. COLLEGE OF ENGINEERING
Bengaluru - 560019, Karnataka, India.

B. M. S. COLLEGE OF ENGINEERING
Bengaluru - 560019, Karnataka, India.

BONAFIDE CERTIFICATE

This is to certify that the Project-3 report titled, **"Yoga Pose Corrector Using Computer Vision"** is a bonafide record of the project work done for the course work **Project-3** by the following students during the academic year 2024-2025.

Student Name	USN	Student's Signature
Lakshmi H Siddaveer	1BM21MD019	
Seggu Harshavardhan	1BM21MD032	
Varuni Gorkalkar	1BM21MD042	

Head of the Department	Guide
Signature: Name: Dr. Jayagowri R Seal: Date:	Signature: Name: Dr. Beena Ullala Mata B N Designation: Professor Date:

Submitted for the University Examination held on ……………..

Internal Examiner: Signature: Name: Designation: Date:	External Examiner: Signature: Name: College/Designation: Date:

B. M. S. COLLEGE OF ENGINEERING
Bengaluru - 560019, Karnataka, India.

CERTIFICATE FROM THE EXTERNAL GUIDE

This is to certify that the Project-3 report titled, **“Yoga Pose Corrector using Computer Vision”** is a bonafide record of the project work done for the course work **Project-3** by the following students during the academic year 2024-2025.

Student Name	USN
Lakshmi H Siddaveer	1BM21MD019
Seggu Harshavardhan	1BM21MD032
Varuni Gorkalkar	1BM21MD042

Guide
Signature:

Name: Sharadha Shetty MS(CS)
Designation: Founder, Sampoorna Yoga
Date:

ACKNOWLEDGEMENT

This project has been successful mainly due to the support from our project guide, Dr. Beena Ullala Mata B N who consistently encouraged us to give our best, particularly during moments of system lags when solutions seemed out of reach. We are grateful to Sharadha Shetty for her guidance in defining the problem statement for this project as our external guide. We are also highly thankful to the Department of Medical Electronics and B. M. S. College of Engineering for giving us the opportunity to research yoga pose correctors and the problems caused by improper practice. This experience has enriched our practical knowledge in the field of computer vision to a great extent.

ABSTRACT

Yoga is one of the most important forms of exercise there is. Incorrect posture while performing asanas may lead to injuries and cramps, or may result in no benefits to the practitioner.

Since yoga is proved to be effective for improving mental and physical health, the instructors usually have 30-60 students per batch, this makes it hard for the instructor to provide one-on-one attention. Hence, to aid the instructor we propose this application where we deploy ML pipeline built on mediapipe pose landmarker that measures the accuracy of the pose and instructs the user to correct it thus reducing the load on the instructor.

The application aims to address the challenges posed by incorrect yoga practices, which can lead to physical strain or injury. By leveraging advancements in computer vision, this system provides real-time feedback to users to ensure proper alignment and posture during yoga sessions. The application is designed to enhance user experience by integrating features such as pose detection, correction, and reference images for guidance. It also incorporates functionality to allow users to seamlessly navigate between poses and exit to the main menu or application as needed. Additionally, the system offers a full-screen display for an immersive experience. This project not only contributes to promoting safe and effective yoga practice but also advances practical applications of computer vision in healthcare and wellness.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1

INTRODUCTION

Overview

Yoga has long been recognized as a holistic practice that fosters both physical and mental well-being. It encompasses a range of asanas (postures) designed to enhance flexibility, strength, and mindfulness. However, the effectiveness of yoga heavily relies on the accuracy of postures, as incorrect alignments can lead to injuries, cramps, or diminished benefits. In traditional yoga sessions, instructors often manage large groups of students, typically ranging from 30 to 60 per batch. This limits the ability to provide individualized attention, making it challenging to correct each student's posture effectively.

Motivation

To address this gap the project deploys a machine learning pipeline built on the Mediapipe Pose Landmarker, the application offers real-time feedback on pose accuracy, guiding users to achieve proper alignment. This not only reduces the instructor's workload but also ensures a safer and more beneficial yoga experience for practitioners.

The application is designed with user convenience in mind, integrating features such as pose detection and correction, along with reference images to assist users during their practice. Additional functionalities include seamless navigation between poses and an option to return to the main menu or exit the application. A full-screen display mode further enhances the immersive experience for users.

By combining the principles of yoga with advancements in computer vision and machine learning, this project aims to make yoga more accessible, accurate, and safe, while also showcasing the potential of technology in healthcare and wellness applications.

CHAPTER 2

LITERATURE SURVEY

A. Chaudhari et al [1] used machine learning techniques to correct pose. Aimed at providing concise feedback for pose improvement and to prevent injuries. Deep learning model using CNN for yoga pose classification along with human joints localisation model was done. They achieved a classification accuracy of 95% for pose identification.

DM Kishore et al [2] uses a camera to capture the picture of the person performing the yoga asana. The captured picture is compared with the benchmark postures. A pretrained deep learning model classifies the pose using a standard dataset. According to the classification the user's pose will be corrected using a voice message to move certain body parts in specific direction.

HT Chen et al [3] it integrates computer vision techniques and analyses the practitioner's posture from both front and side views by extracting body contour, skeleton, dominant axes and feature points. Next according to the domain knowledge of yoga training, visualised instructions for posture rectification are presented so that the user can adjust their posture.

Fujiyoshi et al [4] this paper analysed motion of human target in a video stream by detecting their boundaries. A "star" skeleton is produced from that. Two motion cues are determined from this skeletonization: body posture and cyclic motion of skeleton segments. These cues are used to determine human activities. It doesn't require an *a priori* human model or a large number of pixels on target and it is computationally inexpensive.

Y. Tomar et al [5] this paper discusses the use of OpenCV and Mediapipe in an application to track body movements. It is designed to detect and track key points of body movement using real-time video streaming by webcam. It has capability to check and correct body pose by specifying angles between body part. Main goal is to detect, monitor and identify the actions of individual through video streaming and then produce results by showing it on the same screen as that of a video.

Wagh V et al [6] discuss the use of markerless motion tracking methods for use in a range of domains including clinical settings where traditional marker-based systems fro human pose estimation is not feasible.

CHAPTER 3

AIM AND SCOPE

Aim of this project is to develop a computer vision application using python as an aid to yoga instructors handling large batch sizes of students to help in correcting their pose's accuracy without missing any student and thereby preventing unnecessary injuries to them.

Scope of this project is divided into following categories:

Real-Time Feedback

Provides immediate pose correction suggestions using a machine learning pipeline based on Mediapipe Pose Landmarker, ensuring proper posture and alignment during yoga sessions.

Enhanced Accessibility

Enables yoga practitioners to practice independently with minimal supervision, making yoga more accessible to individuals without direct access to professional instructors.

Instructor Assistance

Supports yoga instructors by reducing their workload in group settings, allowing them to focus on guiding multiple students effectively.

User-friendly features

Incorporates intuitive navigation for users to switch between poses, exit to the main menu, or end the session easily.

Displays reference images alongside real-time feedback to guide users in achieving the correct pose.

Minimal UI for navigation between poses has been implemented using python.

CHAPTER 4

METHODOLOGY

Problem Identification

Defined the problems faced by yoga instructors in handling large batches (more than 30 students at a time) which leads to poor attention to some students which eventually causes poor posture which inevitably causes injuries. Yoga beginners, for whom reaching a professional yoga instructor is not feasible for them this app will be helpful as well.

Research and Technology Selection

Research conducted is highlighted in literature review. Based on the literature review, the technologies selected is as follows: Mediapipe open-source landmark detection frame work for pose detection, python's OpenCV library for integration and development.

Pose Detection and Evaluation

Developed the framework with hard-coded angle values for each joint in each pose. The mediapipe landmarks were used to calculate the angles between the joints in real time for each pose and compare it with the hard coded values. Feedback is giving by text-to-speech feature. Multi-threading was done to solve the problem of lagging in the application.

Application Development

Frontend Development

Designed an intuitive basic UI with features as follows:

- Real-time feedback display
- Reference image for guidance
- Options to navigate between poses and to exit the application

Backend Development

- Integrated the pose detection pipeline
- Implement logic for error detection and correction suggestions
- Optimize for real-time performance by doing multi-threading

Code Development

```
import cv2
import mediapipe as mp
import numpy as np
import pyttsx3
import threading
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox

# Initialize MediaPipe Pose
mp_pose = mp.solutions.pose
pose = mp_pose.Pose()
mp_drawing = mp.solutions.drawing_utils

# Initialize text-to-speech engine
engine = pyttsx3.init()

# Multithreading for TTS
def speak_async(text):
    def run():
        engine.say(text)
```

```
        engine.runAndWait()
    t = threading.Thread(target=run)
    t.start()

# Function to calculate angle between three points
def calculate_angle(a, b, c):
    a = np.array(a)  # First point
    b = np.array(b)  # Mid point
    c = np.array(c)  # End point

    radians = np.arctan2(c[1] - b[1], c[0] - b[0]) - np.arctan2(a[1] -
b[1], a[0] - b[0])
    angle = np.abs(radians * 180.0 / np.pi)

    if angle > 180.0:
        angle = 360 - angle

    return angle

# Function to check if pose matches reference angles
def check_pose(live_angles, reference_pose):
    for joint, ref_angle in reference_pose.items():
        if not (ref_angle - 10 <= live_angles[joint] <= ref_angle + 10):
# ±10° tolerance
            return False, joint
    return True, None

# Predefined reference angles and feedback for poses
poses = {
    "Tree Pose": {
        "reference_angles": {
            "knee_angle": 90,  # Bent leg
            "left_elbow_angle": 160,  # Raised arms
            "right_elbow_angle": 160,  # Raised arms
        },
        "feedback": {
            "knee_angle": "Bend your knee more.",
            "left_elbow_angle": "Raise your left arm higher.",
            "right_elbow_angle": "Raise your right arm higher.",
        },
        "image": "images/tree_pose.png",  # Reference image for Tree Pose
    },
    "Warrior Pose": {
        "reference_angles": {
            "front_knee_angle": 90,  # Front knee
            "back_leg_angle": 180,  # Back leg straight
            "left_elbow_angle": 160,
            "right_elbow_angle": 160,
        },
        "feedback": {
            "front_knee_angle": "Bend your front knee to 90 degrees.",
```

```
            "back_leg_angle": "Straighten your back leg.",
            "left_elbow_angle": "Raise your left arm higher.",
            "right_elbow_angle": "Raise your right arm higher.",
        },
        "image": "images/warrior_pose.png",  # Reference image for Warrior
Pose
    },
    "Chair Pose": {
        "reference_angles": {
            "knee_angle": 90,  # Knees bent at 90 degrees
            "left_elbow_angle": 160,
            "right_elbow_angle": 160,
        },
        "feedback": {
            "knee_angle": "Bend your knees further.",
            "left_elbow_angle": "Raise your left arm higher.",
            "right_elbow_angle": "Raise your right arm higher.",
        },
        "image": "images/chair_pose.png",  # Reference image for Chair
Pose
    },
}

# GUI Implementation
class YogaPoseApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Yoga Pose Corrector")
        self.selected_pose = tk.StringVar(value="Select Pose")

        # Main GUI Layout
        tk.Label(root, text="Yoga Pose Corrector", font=("Helvetica", 18,
"bold")).pack(pady=10)

        # Dropdown for pose selection
        tk.Label(root, text="Choose a Pose:").pack()
        self.pose_menu = ttk.Combobox(root,
textvariable=self.selected_pose, state="readonly")
        self.pose_menu['values'] = list(poses.keys())
        self.pose_menu.pack(pady=5)

        # Start button
        self.start_button = tk.Button(root, text="Start Pose Correction",
command=self.start_pose_correction)
        self.start_button.pack(pady=10)

        # Quit button
        self.quit_button = tk.Button(root, text="Quit",
command=self.quit_app)
        self.quit_button.pack(pady=10)
```

```
    def start_pose_correction(self):
        pose_name = self.selected_pose.get()
        if pose_name == "Select Pose":
            messagebox.showwarning("Selection Error", "Please select a
pose to continue.")
            return

        # Call the pose-correction function
        self.run_pose_correction(pose_name)

    def run_pose_correction(self, selected_pose):
        reference_pose = poses[selected_pose]["reference_angles"]
        feedback_messages = poses[selected_pose]["feedback"]
        reference_image_path = poses[selected_pose]["image"]

        # Load reference image
        reference_image = cv2.imread(reference_image_path)
        if reference_image is None:
            speak_async("Reference image not found. Please check the file
path.")
            return

        # Start camera feed
        cap = cv2.VideoCapture(0)
        cv2.namedWindow("Yoga Pose Corrector", cv2.WINDOW_NORMAL)
        cv2.setWindowProperty("Yoga Pose Corrector",
cv2.WND_PROP_FULLSCREEN, cv2.WINDOW_FULLSCREEN)

        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break

            # Process the frame
            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            results = pose.process(frame_rgb)

            if results.pose_landmarks:
                landmarks = results.pose_landmarks.landmark

                # Define landmark groups
                keypoints = {
                    "left_knee":
[landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].x,

landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].y],
                    "left_hip":
[landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x,

landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].y],
```

```
                    "left_ankle":
[landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value].x,

landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value].y],
                    "right_knee":
[landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].x,

landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].y],
                    "right_hip":
[landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].x,

landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].y],
                    "right_ankle":
[landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value].x,

landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value].y],
                    "right_elbow":
[landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].x,

landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].y],
                    "right_shoulder":
[landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].x,

landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].y],
                    "right_wrist":
[landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value].x,

landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value].y],
                    "left_elbow":
[landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,

landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y],
                    "left_shoulder":
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,

landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y],
                    "left_wrist":
[landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,

landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].y],
                }

                # Calculate angles based on the selected pose
                live_angles = {}
                try:
                    if "knee_angle" in reference_pose:
                        live_angles["knee_angle"] =
calculate_angle(keypoints["left_hip"], keypoints["left_knee"],
keypoints["left_ankle"])
                    if "left_elbow_angle" in reference_pose:
```

```
                        live_angles["left_elbow_angle"] =
calculate_angle(keypoints["left_shoulder"], keypoints["left_elbow"],
keypoints["left_wrist"])
                    if "right_elbow_angle" in reference_pose:
                        live_angles["right_elbow_angle"] =
calculate_angle(keypoints["right_shoulder"], keypoints["right_elbow"],
keypoints["right_wrist"])

                    # Check if pose matches
                    match, joint = check_pose(live_angles, reference_pose)

                    if match:
                        feedback_text = "Pose is correct! Hold steady."
                        speak_async(feedback_text)
                    else:
                        feedback_text = feedback_messages[joint]
                        speak_async(feedback_text)

                    # Draw feedback text on the frame
                    cv2.putText(frame, feedback_text, (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv2.LINE_AA)

                except KeyError as e:
                    speak_async(f"Missing data for {str(e)}")

                # Draw landmarks on the frame
                mp_drawing.draw_landmarks(frame, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS)

            # Show the reference image on the side
            reference_resized = cv2.resize(reference_image, (200, 200))
            frame[10:210, 10:210] = reference_resized

            # Display the frame
            cv2.imshow("Yoga Pose Corrector", frame)

            # Exit on 'q' key press
            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

        # Release resources
        cap.release()
        cv2.destroyAllWindows()

    def quit_app(self):
        self.root.destroy()

# Main application
if __name__ == "__main__":
    root = tk.Tk()
    app = YogaPoseApp(root)
```

```
root.mainloop()
```

Testing and Debugging

Tested by 10 classmates. Received feedback that the program would lag and sometimes crash mid pose detection.

Debugged this by separating the capture code and process code into separate threads and text-to-speech on another separate thread. This stopped the lagging and crashing mid pose detection.

Added reference image to be displayed on screen during pose detection and correction to help the user.

CHAPTER 5

RESULTS AND DISCUSSIONS

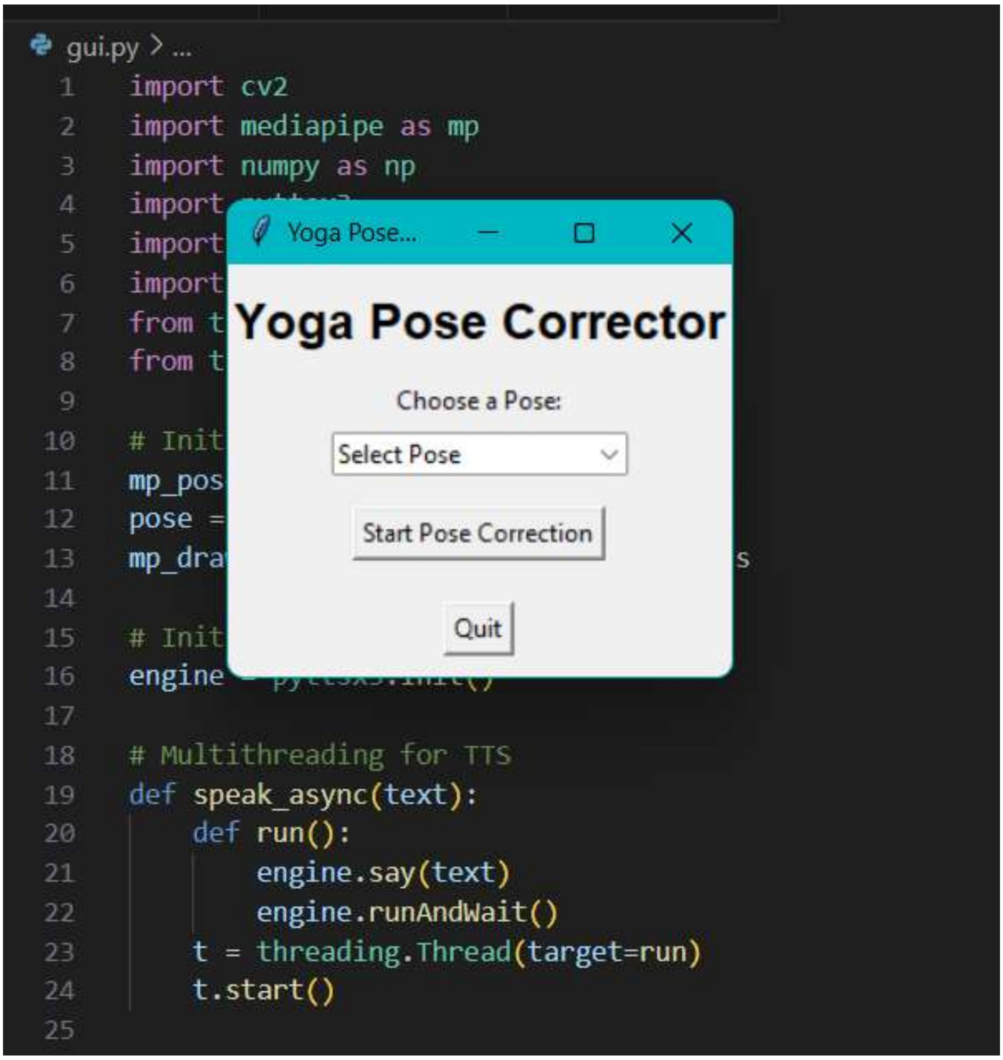

Figure 1: Menu to select pose and begin

Figure 2: Correction message

Figure 3: Pose match, hold steady message

CHAPTER 6

CONCLUSION

The yoga pose corrector project successfully addresses a critical gap in the practice of yoga by offering a technological solution to ensure safe and effective posture alignment. Through the integration of advanced computer vision techniques, specifically the Mediapipe Pose Landmarker, this application provides real-time feedback to users, helping them correct their poses and minimize the risks associated with improper alignment. This not only enhances the individual practitioner's experience but also serves as a valuable tool for yoga instructors, enabling them to manage large groups more efficiently.

The application is designed with user-centric features, such as pose detection, corrective feedback, and reference images for guidance, making it an accessible and practical tool for yoga practitioners of all levels. The inclusion of a seamless navigation system, full-screen mode, and intuitive user interface ensures an immersive and engaging experience. Furthermore, the application demonstrates the immense potential of computer vision and machine learning technologies in promoting health and wellness, paving the way for future innovations in this domain.

The project also highlights the importance of leveraging technology to bridge the gap between traditional practices and modern needs. By automating the process of pose correction, this application not only reduces the dependency on instructors but also ensures that users can practice yoga independently with confidence.

In conclusion, the yoga pose corrector project represents a significant step forward in blending traditional wellness practices with cutting-edge technology. It offers a scalable, efficient, and user-friendly solution that not only enhances the practice of yoga but also promotes the broader adoption of technology in health and wellness applications.

CHAPTER 7

SCOPE FOR FUTURE WORK

The feedback in the current system is hard-coded, based on whether the calculated angles from the landmark points match predefined values. This approach needs enhancement by incorporating generative AI, enabling more dynamic, non-monotonous, and accurate feedback.

Testing should also be conducted in real yoga class settings to evaluate logistics such as camera placement, laptop positioning, and space requirements, with the goal of refining the overall user experience.

In future iterations, the program should be optimized to function effectively under poor internet connectivity, ensuring accessibility and reliability in diverse environments.

The existing UI of the app is very minimal. In future version the UI can be improved to more accessibility friendly.

User data security protocols need to be put into practice in future work.

REFERENCES

[1] A. Chaudhari, O. Dalvi, O. Ramade and D. Ambawade, "Yog-Guru: Real-Time Yoga Pose Correction System Using Deep Learning Methods," 2021 International Conference on Communication information and Computing Technology (ICCICT), Mumbai, India, 2021, pp. 1-6, doi:10.1109/ICCICT50803.2021.9509937.

[2] Kishore, D Mohan; Bindu, S1; Manjunath, Nandi Krishnamurthy. Smart Yoga Instructor for Guiding and Correcting Yoga Postures in Real Time. International Journal of Yoga 15(3):p 254-261, Sep–Dec 2022. | DOI: 10.4103/ijoy.ijoy_137_22

[3] Chen, HT., He, YZ. & Hsu, CC. Computer-assisted yoga training system. Multimed Tools Appl 77, 23969–23991 (2018). https://doi.org/10.1007/s11042-018-5721-2

[4] Fujiyoshi, H., Lipton, A.J. and Kanade, T., 2004. Real-time human motion analysis by image skeletonization. IEICE TRANSACTIONS on Information and Systems, 87(1), pp.113-120.

[5] Y. Tomar, Himanshu, S. Devi and H. Kaur, "Human Motion Tracker using OpenCv and Mediapipe," 2023 3rd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bengaluru, India, 2023, pp. 1199-1204, doi: 10.1109/ICIMIA60377.2023.10425865. keywords: {Performance evaluation;Tracking;Webcams;Streaming media;Real-time systems;Monitoring;Sports;Gesture;OpenCV;Media-Pipe;Video streaming},

[6] Wagh V, Scott M, Kraeutner S, “Quantifying Similarities Between MediaPipe and a Known Standard to Address Issues in Tracking 2D Upper Limb Trajectories: Proof of Concept Study”, JMIR Form Res 2024;8:e56682, URL: https://formative.jmir.org/2024/1/e56682, DOI: 10.2196/56682

PO ATTAINMENT

POs met	PROGRAM OUTCOMES (POs)	Justification
PO 1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems	Used mathematics knowledge to calculate angles made at the joints.
PO 2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences	Literature review on yoga pose correction using machine learning techniques was done.
PO 3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.	Machine learning pipelines were built for the pretrained Mediapipe framework, UI was designed in a minimalistic style for the menu to select the poses.
PO 4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.	Research based knowledge used was that of comparing mediapipe framework and a known standard in a paper.

PO 5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations	Usage of python language, computer vision framework, mediapipe.
PO 6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.	Used to avoid user's from injuring themselves while doing yoga poses.
PO 7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.	It doesn't have any direct impact on environmental aspect as it is a software application.
PO 8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.	Data safety is to be addressed in future work.
PO 9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.	Team members have contributed equally in both research and development of this project.

PO 10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.	Effectively communicated with yoga instructor handling large batch sizes and discussed to brain storm this project's problem statement.
PO 11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.	There were no costs incurred as the team successfully managed to find open-source resources throughout the project.
PO 12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change	The team was actively involved in learning new technologies that was needed for this project from computer vision to multithreading the code.