PhysioBuddy: A Real-World Django-React System for Remote Physiotherapy Pose Assessment

Prof. Meera Sawalkar, Atharva Wattamwar, Sailesh Bhoite, Kalpesh Gawas, Omkar Gite

B.E. Computer Department, JSPM Narhe Technical Campus

---***---

Abstract - PhysioBuddy, an AI-driven web application. This system utilizes state-of-the-art Computer Vision models to perform real-time Human Pose Estimation (HPE) on user video streams, ensuring the correct execution of orthopedic exercises. The primary objective is to provide instant, actionable form correction and repetition counting, thereby maximizing therapeutic efficacy and preventing injury. The methodology centers on a decoupled architecture using Django and React, communicating via WebSockets for low-latency feedback. Results demonstrate the capability to process video frames and return form validation offering a viable and scalable alternative to in-person supervision. The application offers a secure platform where doctors can prescribe exercises and monitor patient compliance and performance.

Key Words: Human Pose Estimation, Real-time Systems, Django Channels, Computer Vision, Physiotherapy, Rehabilitation.

1.INTRODUCTION

Traditional remote supervision relies on delayed video submission or periodic in-person sessions, both of which are resource-intensive and fail to provide immediate corrective action. PhysioBuddy targets this need by integrating real-time pose estimation with a web application that supports clinician-patient workflows. This project introduces a system designed to deliver continuous, real-time feedback using easily accessible consumer hardware (a standard webcam). This approach democratizes expert-level guidance, making it available at home and drastically improving patient adherence and safety. The framework is built to demonstrate the integration of machine learning inference on a web server with low-latency client interaction. This template is intended to be a tool to improve manuscript clarity for the reviewers.

2. BODY OF PAPER

The body of the paper consists of numbered sections that present the methodology. The system architecture is segmented into three primary layers: the Frontend (React), the Backend API and Business Logic (Django), and the Communication Layer (WebSockets).

2.1 Front-end Implementation (React)

The front-end, developed using the React framework, handles the video capture and the display of real-time metrics. The **navigator.mediaDevices.getUserMedia()** JavaScript API is utilized to access the webcam feed. Frames are captured from an HTML canvas and immediately converted into a base64-encoded string for transmission. This process is repeated at a high frequency (e.g., 10-20 frames per second) to maintain a continuous stream.

2.2 Backend Logic and Pose Estimation

The core intelligence resides in the Django backend. The server uses **OpenCV (Open-Source Computer Vision Library)** to decode the incoming base64 video frames into numerical array format. These arrays are passed to the
MediaPipe Pose model, a state-of-the-art solution for **Human Pose Estimation (HPE)**. The model returns a set of key points (landmarks) for the user's body joints. Business logic then applies biomechanical rules (angles, distance ratios) to these landmarks to determine form correctness and calculate repetition counts. Sec. 2.1 contained a description of the front-end capture.

2.3 Real-time Communication (WebSockets)

The communication layer utilizes **Django Channels** to enable a persistent **WebSocket** connection between the React client and the Django server. This is critical for achieving the low latency required for real-time feedback. The communication flow is entirely data-centric: the client sends the frame data as JSON, and the server responds instantly with a JSON payload containing the calculated repetition count and corrective feedback messages. This architectural choice bypasses the limitations of the traditional request-response (HTTP) model. The project uses a decoupled approach, with the main API views implemented without relying on Django REST Framework (DRF) for initial deployment, focusing on custom JsonResponse objects for efficiency.

2.4 Data Management and User Roles

The system employs a relational database (MySQL in production) to manage two primary user roles: **Doctor** and **Patient**. The built-in Django User model is extended using the '*OneToOneField*' relationship to create distinct '*DoctorProfile*' and '*PatientProfile*' models. Doctors, identified by the '*is_staff*' attribute, access a admin panel for assigning exercises, while Patients interact solely with the dedicated web interface.

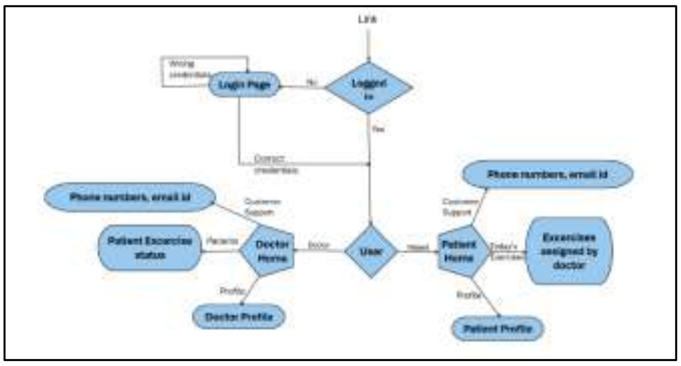

Fig-1: User Flow Diagram of PhysioBuddy showing login and role-based navigation.

3. CONCLUSIONS

PhysioBuddy is a modular, secure, and pragmatic prototype for remote physiotherapy support. It integrates real-time pose estimation with a Django-backed web service to provide immediate patient feedback and clinician accessible session records. The architecture supports incremental improvements such as model personalization, expanded exercise templates, and integration with teleconferencing systems for supervised sessions.

ACKNOWLEDGEMENT

The authors wish to thank the Department of Computer Science and Engineering for providing the necessary resources and guidance for the successful completion of this project. The authors acknowledge any contributors, testers, and early clinicians who provided feedback during prototype development.

REFERENCES

1) Yafeng Zhou, Fadilla 'Atyka Nor Rashid, Marizuana Mat Daud, Mohammad Kamrul Hasan and Wangmei Chen: Machine Learning-Based Computer Vision for Depth-Camera Physiotherapy Movement Assessment (2025): https://www.mdpi.com/1424-8220/25/5/1586

2) Vaidehi Wagh, Matthew W Scott1, Sarah N Kraeutner1: Quantifying Similarities Between MediaPipe and a Motion-Capture Standard (JMIR Formative, 2024): https://formative.jmir.org/2024/1/e56682/PDF