A PROJECT REPORT

ON

SMART POSTURE ESTIMATION FOR HEALTHCARE USING DEEP LEARNING

A report Submitted in the partial fulfilment of the requirements for the award of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

SUBMITTED BY

KUPPUSWAMY UDAYKIRAN 21BK1A6663

MACHA HARSHA VARDHAN 21BK1A6664

GADAPA NAVEEEN KUMAR 21BK1A6642

UNDER THE ESTEEMED GUIDANCE OF

Ms. Sampa Biswas

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI&ML)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI&ML)

St. Peter's Engineering College (UGC Autonomous)

Approved by AICTE, New Delhi, Accredited by NBA and NAAC with 'A' Grade,

Affiliated to JNTU, Hyderabad, Telangana.

2024-2025

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI&ML)

CERTIFICATE

This is to certify that project entitled **"Smart Posture Estimation for Healthcare using Deep Learning"** is carried out by **KUPPUSWAMY UDAY KIRAN (21BK1A6663), MACHA HARSHA VARDHAN (21BK1A6664), GADAPA NAVEEN KUMAR (21BK1A6642)** in partial fulfilment for the award of the degree of **Bachelor of Technology in COMPUTER SCIENCE & ENGINEERING (AI&ML)** is a record of bonafide work done by them under my supervision during the academic year "2024–2025".

INTERNAL GUIDE
Ms. Sampa Biswas
Assistant Professor
Department of CSE(AI&ML)
St. Peter's Engineering College (A)
Hyderabad- 500043.

HEAD OF THE DEPARTMENT
Dr. P. Deepan
Associate Professor & Head
Department of CSE(AI&ML)
St. Peter's Engineering College (A)
Hyderabad- 500043.

PROJECT COORDINATOR
Dr. P. V. Kishore
Professor
Department of CSE(AI&ML)
St. Peter's Engineering College (A)
Hyderabad- 500043.

EXTERNAL EXAMINER

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

ACKNOWLEDGEMENT

We sincerely express our deep sense of gratitude to **Ms. Sampa Biswas**, for her valuable guidance, encouragement and cooperation during all phases of the project.

We are greatly indebted to our Project Coordinator **Dr. P.V. Kishore**, for providing valuable advice, constructive suggestions and encouragement without whom it would not been possible to complete this project.

It is a great opportunity to render our sincere thanks to **Dr. P. Deepan**, Head of the Department, CSE (Artificial Intelligence & Machine Learning) for his timely guidance and highly interactive attitude which helped us a lot in successful execution of the Project.

We are extremely thankful to our Principal **Dr. N. Chandra Sekhar Reddy**, who stood as an inspiration behind this project and heartfelt for his endorsement and valuable suggestions.

We respect and thank our Administrative Director **Mr. T. Anuraag Reddy**, Academic Director **Mrs. T. Saroja Reddy** and Secretary **Sri. T. V. Reddy**, for providing us an opportunity to do the project work at **St. PETER'S ENGINEERING COLLEGE** and we are extremely thankful to them for providing such a nice support and guidance which made us to complete the project.

We also acknowledge with a deep sense of reverence, our gratitude towards our parents, who have always supported us morally as well as economically. We also express gratitude to all our friends who have directly or indirectly helped us to complete this project work.

KUPPUSWAMY UDAY KIRAN 21BK1A6663

MACHA HARSHA VARDHAN 21BK1A6664

GADAPA NAVEEN KUMAR 21BK1A6642

INSTITUTE VISION

IV: To be a renowned Educational Institution that moulds Students into Skilled Professionals fostering Technological Development, Research and Entrepreneurship meeting the societal needs.

INSTITUTE MISSION

IM1: Making students knowledgeable in the field of core and applied areas of Engineering to innovate Technological solutions to the problems in the Society.

IM2: Training the Students to impart the skills in cutting edge technologies, with the help of relevant stake holders.

IM3: Fostering conducive ambience that inculcates research attitude, identifying promising fields for entrepreneurship with ethical, moral and social responsibilities.

DEPARTMENT VISION

DV: To improve fundamentals and make it easier to address the ever expanding needs of society by producing the best leaders in artificial intelligence and machine learning via excellence in research and education.

DEPARTMENT MISSION

DM1: Create centers of excellence in cutting-edge computing and artificial intelligence.

DM2: Impart rigorous training to generate knowledge through the state-of-the-art concepts and technologies in Artificial Intelligence and Machine Learning.

DM3: To enhance research in emerging areas by collaborating with industries and institutions at the national and international levels.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Work effectively in inter-disciplinary field with the knowledge of Artificial Intelligence and Machine Learning to develop solutions to the real-world problems.

PEO2: To communicate and work effectively on team based engineering projects and will practice the ethics of their profession consistent with a sense of social responsibility.

PEO3: Excel as socially committed engineers or entrepreneurs with high ethical and moral values.

PROGRAM OUTCOMES (POs)

PO1- Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and provide solutions in the engineering specialization of artificial intelligence and machine learning.

PO2- 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.

PO3- 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, and environmental considerations.

PO4- 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.

PO5- Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6- 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.

PO7- 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.

PO8- Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9- Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10- 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.

PO11- 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.

PO12- 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.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Apply fundamental concepts of Artificial Intelligence and Machine Learning to solve multidisciplinary engineering problems.

PSO2: To communicate and work effectively on team based engineering projects and will practice the ethics of their profession consistent with a sense of social responsibility.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

DECLARATION

We declare that a Major Project entitled **"Smart Posture Estimation for Healthcare using Deep Learning"** is an original work submitted by the following group members who have actively contributed and submitted in partial fulfillment for the award of degree in **"Bachelor of Technology in Computer Science and Engineering (AI&ML)"**, at **St. Peter's Engineering College, Hyderabad,** and this project work has not been submitted by me to any other college or university for the award of any kind of degree.

Group No: A10

Program: B. Tech

Branch: CSE(AI&ML)

Major Project Title: Smart Posture Estimation for Healthcare using Deep Learning

Date Submitted:

KUPPUSWAMY UDAY KIRAN	**21BK1A6663**
MACHA HARSHA VARDHAN	**21BK1A6664**
GADAPA NAVEEN KUMAR	**21BK1A6642**

ABSTRACT

This project delves into the application of artificial intelligence and computer vision for real-time posture analysis, aiming to combat the growing concerns surrounding sedentary lifestyles and poor ergonomic habits. Leveraging deep learning models trained on skeletal keypoint data, the system is designed to detect postural misalignments during activities such as prolonged sitting and standing. By capturing and analyzing body landmarks through pose estimation, the model identifies deviations from ideal alignment and provides tailored feedback to encourage corrective actions. The study employs metrics such as accuracy, precision, recall, and mean squared error to rigorously evaluate model performance. Ethical considerations, particularly around data privacy and the system's intended use as a supportive health tool rather than a diagnostic replacement, are also critically addressed. The findings underscore the promising potential of AI in recognizing harmful posture patterns, offering a proactive solution to reduce the risk of musculoskeletal disorders and enhance ergonomic health in day-to-day life.

Furthermore, the project integrates pre-trained neural network architectures and publicly available skeletal datasets to create a robust training environment for detecting poor posture. The system's ability to classify common alignment issues—such as slouching, forward head posture, and uneven shoulder alignment—was tested and refined across diverse samples. The most effective model exhibited high performance in identifying misalignments, showcasing the practicality of AI-powered posture monitoring tools. Importantly, this emphasizes that while technology can provide real-time feedback and promote awareness, long-term posture improvement requires an integrative approach that includes ergonomic practices and behavioral adjustments. By demonstrating the feasibility of intelligent posture assessment systems, this study lays the groundwork for future innovations in digital health monitoring, potentially extending into workplace ergonomics, rehabilitation, and preventive care.

Keywords: Posture analysis, Skeletal keypoints, computer vision, ergonomic health, AI in healthcare

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER- I

INTRODUCTION

1.1 INTRODUCTION

Maintaining correct posture is essential for ensuring long-term musculoskeletal health, joint stability, and overall physical well-being. However, in the modern digital era, posture-related health problems have seen a significant rise, primarily due to increasingly sedentary lifestyles. Many individuals now spend long hours working on computers, using smartphones, or sitting in ergonomically unsupportive environments—whether at home or in the workplace. These prolonged and often incorrect postures can lead to a variety of issues including chronic back and neck pain, spinal misalignment, joint stiffness, poor blood circulation, fatigue, and even long-term degenerative conditions. Despite these risks, early signs of postural degradation often go unnoticed. Traditional posture assessments conducted by physiotherapists or healthcare providers typically rely on visual inspection or wearable sensors, which, although useful, are limited by subjectivity, cost, lack of scalability, and the inability to provide real-time feedback or long-term monitoring.

In response to these challenges, artificial intelligence (AI), particularly computer vision and deep learning, has emerged as a powerful tool for posture analysis and correction. AI-based systems can leverage pose estimation frameworks like MediaPipe to track and extract skeletal keypoints from images or videos, enabling the automated detection of misalignments or abnormalities in body posture. When integrated with trained machine learning models, these systems can classify and provide insights on specific posture conditions, offering real-time corrective feedback without the need for human supervision. This technological approach transforms posture assessment from a manual and intermittent process into a continuous, objective, and user-friendly solution. Moreover, these systems can be deployed in various environments—clinics, workplaces, schools, or even homes—through web applications, mobile interfaces, or smart devices.

1.2 OBJECTIVE OF THE PROJECT

The main objective of the Smart Posture Estimation for Healthcare project is to create a reliable, AI-powered solution that can automatically evaluate human posture, identify deviations from ideal alignment, and assist in the prevention and early detection of musculoskeletal disorders. With the widespread adoption of digital devices and the rise of sedentary behavior, poor posture has become a common cause of back pain, joint discomfort, and spinal complications. This project addresses the growing need for an accessible, non-invasive, and cost-effective method of posture evaluation by integrating deep learning and computer vision techniques. Using MediaPipe's pose estimation framework, the system extracts key skeletal landmarks from static images or real-time video feeds and analyzes these points to detect irregularities in posture. If deviations are found, the system generates corrective feedback, allowing users to adjust their positioning promptly—helping reduce the risk of long-term physical strain and promoting better postural habits.

In addition to technical accuracy, the project emphasizes accessibility and usability across different environments. A user-friendly web interface has been developed using Flask, allowing users—from patients and fitness enthusiasts to healthcare professionals and ergonomists—to interact with the system seamlessly. Through this platform, users can upload media, receive automated evaluations, and view real-time insights without the need for advanced hardware or clinical supervision. Moreover, the project extends its objective beyond posture correction by laying the groundwork for early screening of neurological or movement-related disorders such as Parkinson's disease and arthritis. Ethical considerations are also taken into account, ensuring user data privacy, unbiased model predictions, and transparency in feedback. By combining intelligent automation with practical healthcare needs, this project aims to enhance digital wellness tools and support preventive healthcare strategies. Ultimately, it offers a scalable solution that empowers individuals to take control of their physical well-being in everyday settings—at home, at work, or in medical environments.

1.3 ORGANIZATION OF THE THESIS

This thesis is organized into six chapters, each detailing a key phase in the development of the Smart Posture Estimation system. The structure ensures a logical flow from problem identification to final implementation and evaluation.

Chapter 1 provides an overview of the study, including the motivation for developing an AI-based posture analysis system. It outlines the project's objectives, significance, and scope in addressing posture-related health issues caused by sedentary lifestyles.

Chapter 2 reviews relevant literature on posture detection, pose estimation techniques, and AI applications in healthcare. It highlights existing methods, identifies research gaps, and establishes the need for the proposed solution.

Chapter 3 describes the system architecture and methodology, covering components like the MediaPipe-based pose estimation module, classification logic, dataset preparation, and the integration of a Flask-based web interface.

Chapter 4 outlines the implementation process, detailing how images and videos are processed to extract skeletal keypoints and how these are used to evaluate posture. It also explains the functioning of the web interface and real-time feedback system.

Chapter 5 focuses on testing and validation. It discusses the testing strategies, performance metrics (accuracy, precision, recall), and the model's evaluation on real and synthetic data to ensure reliability and robustness.

Chapter 6 concludes the thesis by summarizing the project outcomes, key contributions, and limitations. It also presents future enhancement possibilities, such as mobile deployment, advanced classification, and integration with real clinical data.

1.4 PROPOSED SOLUTION

The proposed system presents a novel, real-time, camera-based posture analysis tool tailored for smart healthcare applications. It is designed to identify and monitor human posture quality using geometric angle calculations and pose estimation techniques, without relying on complex supervised learning models. The system utilizes a lightweight pose tracking framework such as MediaPipe, which extracts 33 key human body landmarks through a webcam or camera feed. From these landmarks, it calculates crucial biomechanical angles like neck inclination, torso deviation, and shoulder alignment using trigonometric relations. These angles are compared against predefined ergonomic thresholds derived from physiotherapy guidelines and biomechanical studies to determine whether the user's posture is good or poor. This threshold-based approach allows for consistent and explainable evaluation without needing labeled training data, ensuring privacy, ease of deployment, and reduced computational overhead.

A key innovation in this system is its real-time graphical interface, developed using Tkinter and OpenCV, which overlays body keypoints on the video feed and presents live metrics on posture health. The display includes intuitive indicators for each angle (marked as "Good" or "Aligned"), posture duration counters, and a live Session Stats graph showing fluctuations between correct and incorrect posture over time. This time-based feedback mechanism enhances user awareness and accountability by tracking how long a user maintains good posture versus slouching. Additionally, the system's logic is modular and extensible, allowing customization of posture types (e.g., standing, sitting, stretching) and ergonomic goals for various users such as office workers, students, or patients undergoing physiotherapy. Since the tool does not depend on external datasets, training cycles, or depth cameras, it can run effectively on low-power devices such as laptops and tablets, enabling widespread use in home or clinical settings.

1.5 ADVANATGES OF PROPOSED SOLUTION

1. High Accuracy in Controlled Environments

Pose estimation models typically perform with high precision when tested in controlled environments with good lighting and minimal occlusion.

2. Ability to Handle Complex Body Postures

Advanced models are capable of detecting and tracking intricate body movements and postures, making them suitable for applications in sports, rehabilitation, and animation.

3. Real-Time Processing Potential

Some recent methods are designed to work in real-time, enabling immediate feedback and making them suitable for interactive applications like fitness monitoring and virtual reality.

4. Multi-Person Detection Capabilities

Many modern approaches are designed to detect and estimate poses for multiple individuals in a single frame, which is useful for group activities or crowd monitoring.

5. Scalability with Robust Datasets

The use of large-scale, diverse datasets enables the models to generalize well across different body types, environments, and clothing styles in various applications.

6. Flexibility Across Modalities

Pose estimation systems can be applied to different input modalities (e.g., RGB images, depth data, or even thermal cameras), offering flexibility in diverse environments and conditions.

CHAPTER- II

LITERATURE SURVEY

2.1 EXISTING WORKS

Human pose estimation plays a vital role in posture assessment systems, particularly in smart healthcare applications. Several deep learning-based techniques have been proposed to enhance the accuracy and efficiency of pose estimation models. This section explores and analyzes five key papers that significantly contribute to the field of 2D and 3D pose estimation.

Sengupta et al. in their paper titled "Synthetic Training for Accurate 3D Human Pose and Shape Estimation in the Wild" proposed STRAPS, a synthetic training pipeline for accurate monocular 3D human pose and shape estimation from RGB images. Their system relies on proxy representations such as silhouettes and 2D joints as inputs to a regression network trained on synthetic SMPL-based data. They introduced the SSP-3D dataset containing sports person images with pseudo-ground-truth SMPL parameters for evaluation. The method outperformed existing models on SSP-3D in shape prediction accuracy, though its effectiveness may reduce under conditions like occlusion or poses not covered in the synthetic data.

Rim et al. in their paper titled "Real-Time Human Pose Estimation Using RGB-D Images and Deep Learning" introduced a method combining RGB and depth (RGB-D) data to improve the precision and robustness of pose estimation. Their approach achieved a mean Average Precision (mAP) of 0.903 and a mean Average Recall (mAR) of 0.938, surpassing RGB-only and depth-only models. The added depth component helped overcome challenges like lighting variation and occlusions, but dependency on depth sensors limits applicability in certain environments.

Rizwan et al. in their paper titled "Neural Network Approach for 2-Dimension Person Pose Estimation with Encoded Mask and Keypoint Detection*"* proposed a convolutional neural network method integrating encoded masks with keypoint detection to enhance 2D human pose estimation. The model improves accuracy over traditional approaches by effectively locating joint positions. However, the method may face hurdles in real-time usage due to computational intensity and reliance on high-quality image inputs.

Kanase et al. in their paper titled *"*Pose Estimation and Correcting Exercise Posture*"* addressed the use of pose estimation to monitor and correct physical exercise postures in real time. By detecting joint deviations, the system gives users corrective feedback to minimize injury risks and improve performance. The study highlighted the system's potential, though factors like occlusions, variable lighting, and clothing inconsistencies can affect its posture detection reliability.

Neff et al. in their paper titled "EfficientHRNet: Efficient and Scalable High-Resolution Networks for Real-Time Multi-Person 2D Human Pose Estimation" presented EfficientHRNet, a high-resolution architecture that maintains detailed spatial features for accurate keypoint detection across multiple persons. Tested on the COCO2017 dataset, it showed strong accuracy with lower computational overhead compared to existing models. Nevertheless, performance may degrade in scenarios with extreme scale variation or densely populated scenes where retaining high-resolution features is more complex.

2.2 LIMITATION OF EXISTING WORKS

Despite significant advancements in human pose estimation techniques, several consistent limitations persist across modern approaches. A detailed analysis reveals the following critical disadvantages:

1. **Limited Real-World Generalization**
   Most models are trained in controlled or synthetic environments, so they often fail in real-world situations with cluttered backgrounds, occlusions, or unusual body movements.
2. **Dependence on Specialized Hardware**
   Several approaches need depth sensors or advanced hardware, making them impractical for use on standard RGB cameras and everyday devices.
3. **High Computational Overhead**
   These systems generally require heavy processing power, which makes real-time execution difficult on mobile devices or low-resource platforms.
4. **Sensitivity to Environmental Conditions**
   Lighting changes, clothing variations, and partial occlusion can affect accuracy, making the models unreliable in uncontrolled environments.
5. **Scalability Issues in Multi-Person Scenes**
   Performance often drops in crowded scenes due to overlapping limbs and varied body scales, limiting their effectiveness in group settings.
6. **Lack of Fairness and Inclusivity**
   Training datasets often lack diversity, leading to biased results when models are applied to people of different body types, skin tones, or clothing styles.
7. **Limited Real-Time Feedback for Posture Correction**
   Many existing systems focus on pose detection rather than real-time correction, offering limited or delayed feedback, which reduces their usefulness for applications like exercise monitoring or rehabilitation.

CHAPTER- III

ANALYSIS

3.1 PROBLEM STATEMENT

In today's fast-paced, technology-driven world, prolonged sitting, screen exposure, and sedentary work habits have led to a dramatic rise in posture-related health issues such as back pain, spinal misalignment, and chronic musculoskeletal disorders. Traditional methods of assessing and correcting posture are largely reliant on manual observation and physical therapy, which can be subjective, time-consuming, and inaccessible to many individuals—especially those in remote or underserved regions. Moreover, early symptoms of poor posture often go unnoticed until they develop into more serious conditions, making timely intervention critical. In clinical environments, therapists and physicians face the challenge of consistently monitoring posture changes in patients over time without an automated, standardized tool. These limitations highlight the growing need for a smart, real-time, and scalable solution that can assist healthcare professionals and individuals in monitoring and improving postural health effectively.

To address these challenges, this project presents a Smart Posture Estimation System for Healthcare using machine learning and pose estimation technologies. The system is built using Python and Flask, integrating MediaPipe for real-time pose detection and OpenCV for image processing. By capturing key body joint positions from uploaded images or live camera input, the model assesses postural alignment and identifies potential deviations from optimal posture. It categorizes the posture into specific conditions (e.g., "Good", "Forward Head", or "Slouched") using a pre-trained machine learning classifier and provides instant visual and textual feedback through a user-friendly web interface. The application also includes educational guidance and visual feedback mechanisms to promote corrective action. This AI-powered tool enhances traditional physiotherapy and ergonomic assessment practices by enabling remote posture evaluation, personalized monitoring.

3.1. SOFTWARE REQUIREMENTS

The software requirements for the Smart Posture Estimation System define the essential components, libraries, and tools necessary to deliver robust posture detection using machine learning and computer vision. The system is primarily built using Python, selected for its versatility and comprehensive ecosystem for AI, data processing, and web development. It uses state-of-the-art libraries like MediaPipe for real-time human pose estimation, and OpenCV for image and video processing. The frontend is designed using Flask, a lightweight web framework, to provide an accessible and interactive user experience for uploading images or videos and displaying results effectively.

Supported Operating Systems

- Windows 10 or above
- Linux (Ubuntu 20.04+ or equivalent distributions)
- macOS (latest stable version)

Programming Language

- Python 3.8 or higher

Technologies and Libraries Used

Python:

- The core development language, offering simplicity and support for machine learning, computer vision, and data handling.

Flask:

- Used for developing the web interface that handles user interaction, file uploads, result rendering, and display of processed outputs.

MediaPipe:

- A powerful, real-time pose estimation framework by Google that detects 33 human body landmarks for accurate skeletal tracking.

OpenCV (cv2):

- Used to handle image preprocessing, frame extraction, resizing, noise filtering, and enhancements that assist in improving pose detection accuracy.

NumPy:

- Facilitates numerical computations and array-based data manipulation required during image preprocessing and pose vector analysis.

Matplotlib & Seaborn:

- Enable graphical visualization of joint landmarks, angles, deviations, and posture trends through plots, charts, and heatmaps.

Pandas:

- Manages and structures data output, allowing for efficient logging of results, reporting, and generation of posture summaries in readable formats.

TensorFlow:

- Provides deep learning capabilities to classify postures and detect abnormalities using trained models, enhancing accuracy and predictive functionality.

Pickle:

- Used to serialize and deserialize trained models so that they can be stored and reused without needing retraining each time the system is initialized.

OS & Webbrowser Modules:

- The OS module enables file path navigation and directory management, while the webbrowser module is used to launch output result pages automatically in the user's default browser.

By combining these software tools and libraries, the system ensures seamless end-to-end processing—from input acquisition to pose estimation, classification, and result visualization. It is capable of operating both in real-time (live webcam) and on offline data (uploaded media files), making it adaptable to different use cases.

3.2. HARDWARE REQUIREMENTS

The hardware requirements for the Smart Posture Estimation for Healthcare System specify the minimum and recommended specifications necessary for the application to operate efficiently. Since the system leverages computationally intensive processes such as real-time pose estimation, image and video processing, and deep learning-based classification, it is crucial that the hardware setup supports smooth execution and rapid analysis with minimal latency. These specifications ensure that users experience a responsive and reliable system, capable of handling both real-time and offline data inputs.

For optimal performance, a multi-core processor is essential to handle frame-by-frame analysis, data parsing, and background computations efficiently. Adequate RAM ensures that the system can manage large datasets and perform real-time pose tracking without lag. Sufficient storage is necessary for saving the pose estimation models, processed outputs, and generated analysis reports. Furthermore, the use of a dedicated GPU is strongly recommended, especially when leveraging deep learning models, as it significantly accelerates the inference and training times compared to CPU-only processing. For environments where live posture detection is desired, peripheral hardware such as a webcam or external camera module becomes essential.

Minimum Hardware Requirements

Component	Specification
Processor	Intel Core i5 (8th Gen) / AMD Ryzen 5 or equivalent
RAM	8 GB or higher
Storage	100 GB HDD or SSD (SSD preferred for better performance)
Graphics	Integrated GPU (suitable for basic pose/image processing)

The minimum hardware configuration outlined above is sufficient for running the basic functionalities of the posture analysis system, such as analysing images or pre-recorded videos, executing lightweight pose estimation tasks, and interacting with the web interface. It is especially suitable for users who aim to utilize the system for educational, demonstration, or non-intensive healthcare monitoring purposes.

Recommended Hardware Requirements

Component	Specification
Processor	Intel Core i7 (10th Gen or above) / AMD Ryzen 7 or higher
RAM	16 GB or higher
Storage	256 GB SSD or higher
Graphics	Dedicated GPU - NVIDIA GeForce GTX 1650 or higher

Optional (But Recommended) Hardware Components

- Webcam or Camera Module: Required for real-time live posture tracking and monitoring.
- High-resolution Monitor: Improves visibility of posture graphs, joint positions, and diagnostic feedback.
- External Storage Device: Useful for backing up datasets, processed media, and analytical results in large-scale deployments.

By adhering to these hardware requirements, the posture estimation system is guaranteed to deliver accurate and timely insights into users' postural health. A high-performance setup not only ensures real-time responsiveness but also improves the accuracy of posture classification models, making the solution scalable for clinical, personal wellness, and fitness applications.

CHAPTER- IV

DESIGN

4.1 INTRODUCTION

The design phase of the "Smart Posture Estimation for Healthcare using Deep Learning" project focuses on constructing a streamlined and responsive posture analysis system. It integrates multiple components such as a pose estimation model, classification logic, data handling mechanisms, and a user-friendly interface—all organized around a web-based architecture using Flask. The system is designed to take real-time or stored video/image input and process it through a series of steps including pose detection using MediaPipe, normalization, feature extraction, and posture classification using a deep learning model.

4.2 SYSTEM ARCHITECTURE

The system architecture for the posture evaluation framework is illustrated in Figure 4.2. This architecture integrates real-time image capturing, deep learning models, and analytical processing to assess human posture effectively.

1. Data Collection & Preprocessing

This module forms the foundation of the system by managing the acquisition and preparation of input data. It supports both live capture via webcam and uploading of stored images or videos using OpenCV. The captured frames are then processed using MediaPipe, which accurately detects human pose landmarks such as shoulders, spine, knees, and hips. These landmarks are crucial for evaluating posture alignment.

To ensure that the input data is consistent and usable for ML analysis, normalization techniques are applied. This includes resizing, scaling, and reformatting pose points into a structured form that matches the expected input format of the deep learning model. The preprocessing step filters noise and stabilizes the data, improving the reliability of the po

Highlights:

- Real-time and offline input support via OpenCV.
- Pose landmark extraction using MediaPipe's holistic model.
- Data normalization for consistency across frames.
- Modular structure for easy extension with additional sensors or data types.

2. Posture Analysis & Classification

Once the pose landmarks are extracted and structured, this module evaluates the data to determine the user's posture quality. It utilizes a pre-trained deep learning model (potentially a CNN, LSTM, or hybrid approach) to classify posture into different categories—such as normal, slouched, forward head, or spinal misalignment. This classification is essential for identifying common postural problems associated with poor ergonomics or spinal issues.

The system is optimized for real-time inference, enabling users to receive immediate feedback based on their current pose. This responsiveness is especially beneficial for rehabilitation exercises or continuous posture monitoring in a clinical or home setting. The model is trained on pose-based features rather than raw images, making it lightweight and focused on joint relationships.

Highlights:

- Real-time classification of standing, sitting, or walking postures.
- Detection of abnormalities like kyphosis, scoliosis, or hunched shoulders.
- Can be retrained or fine-tuned for specific datasets or age groups.
- Integration-ready with alerts or smart notifications.

3. User Interface & Visualization

The UI is designed using Flask's templating engine and HTML/CSS assets to offer a clean and intuitive experience for users. From the main dashboard, users can start webcam capture, upload a file, or view previous results. The system overlays the detected pose skeleton directly on the media using OpenCV visualization tools, highlighting the body joints and angles.

This visual feedback makes it easier for users to understand the areas where their posture might be incorrect. The interface is kept lightweight for fast loading and is structured in a modular way so future components like charts, severity scales, or time-based comparisons can be added with minimal changes.

Highlights:

- Browser-based interaction via Flask web app.
- Real-time skeleton overlay using OpenCV.
- Easy-to-use interface with dynamic posture feedback.
- Progress tracking capabilities (can be expanded with session logging).

4. Recommendation & Report Module

After detecting posture deviations, the system provides actionable feedback aimed at correcting the issues. These recommendations are derived from a mapping between identified posture types and corrective measures commonly suggested by physiotherapists or fitness professionals. The advice may include exercises like chin tucks, shoulder rolls, core strengthening, or reminders to adjust sitting posture.

While the current version displays recommendations within the web interface, future enhancements include the ability to generate downloadable PDF reports. These reports could summarize recent posture analysis sessions, track long-term progress, and include

visual snapshots or graphs. This feature could be particularly useful for clinical use, where users need to share posture assessments with healthcare providers.

Highlights:

- Personalized advice based on detected posture type.
- Includes simple, effective exercises for correction.
- Potential for report generation using tools like ReportLab.
- Extendable for integration with health monitoring platforms or physiotherapy systems.

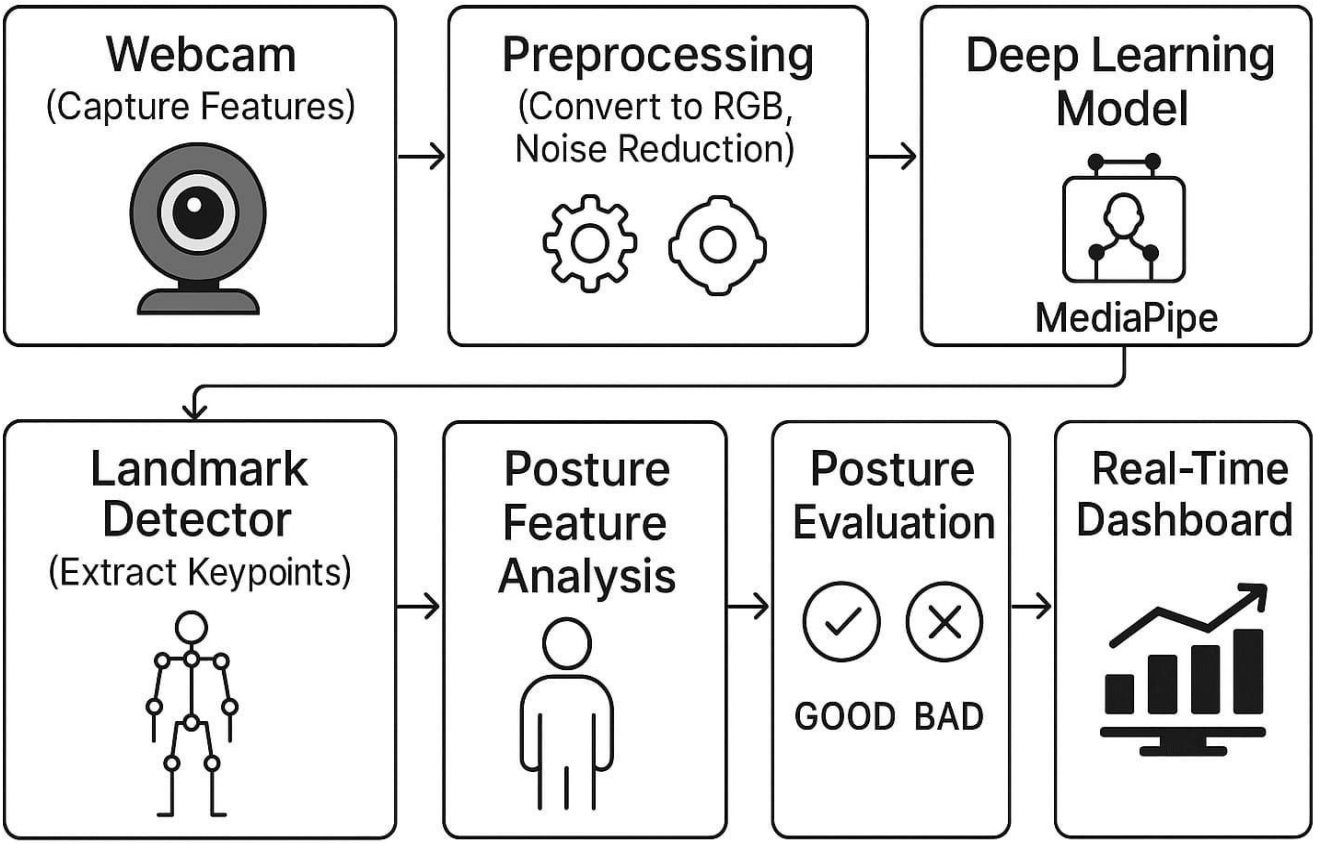

Figure 4.1: System Architecture

4.3 UML Diagrams

UML (Unified Modeling Language) is a standardized modeling language used extensively in object-oriented software engineering. Developed and maintained by the Object Management Group (OMG), UML provides a common visual framework to design, develop, and document the structure and behavior of complex software systems.

At its core, UML is composed of two main elements: a meta-model, which defines the rules and semantics of the language, and notations, which are graphical symbols used to represent software components and their relationships. UML incorporates the best practices of object-oriented analysis and design and has become an essential part of modern software development methodologies.

Goals of UML

The design of UML is driven by several key objectives:

1. To offer developers a standardized and expressive visual language for modeling software systems.
2. To support extensibility, allowing customization and adaptation to specific domains or platforms.
3. To remain independent of programming languages or development methodologies.
4. To provide a formal foundation for understanding and interpreting models.
5. To encourage the evolution of the object-oriented tool ecosystem.
6. To consolidate proven software engineering practices under one unified modeling approach.

4.3.1 Use Case Diagram

The Use Case Diagram is one of the primary behavioral diagrams in UML, used to describe how users interact with a system. For the Smart Posture Estimation System, the Use Case Diagram offers a high-level view of the system's core functionalities and the way users (actors) engage with those functions.

This diagram is especially useful during the design phase as it helps identify system boundaries, user expectations, and required services. It maps out all the primary use cases—such as image or video input, pose analysis, classification, feedback generation, and report viewing—alongside the roles of different actors like end-users, physiotherapists, or administrators.

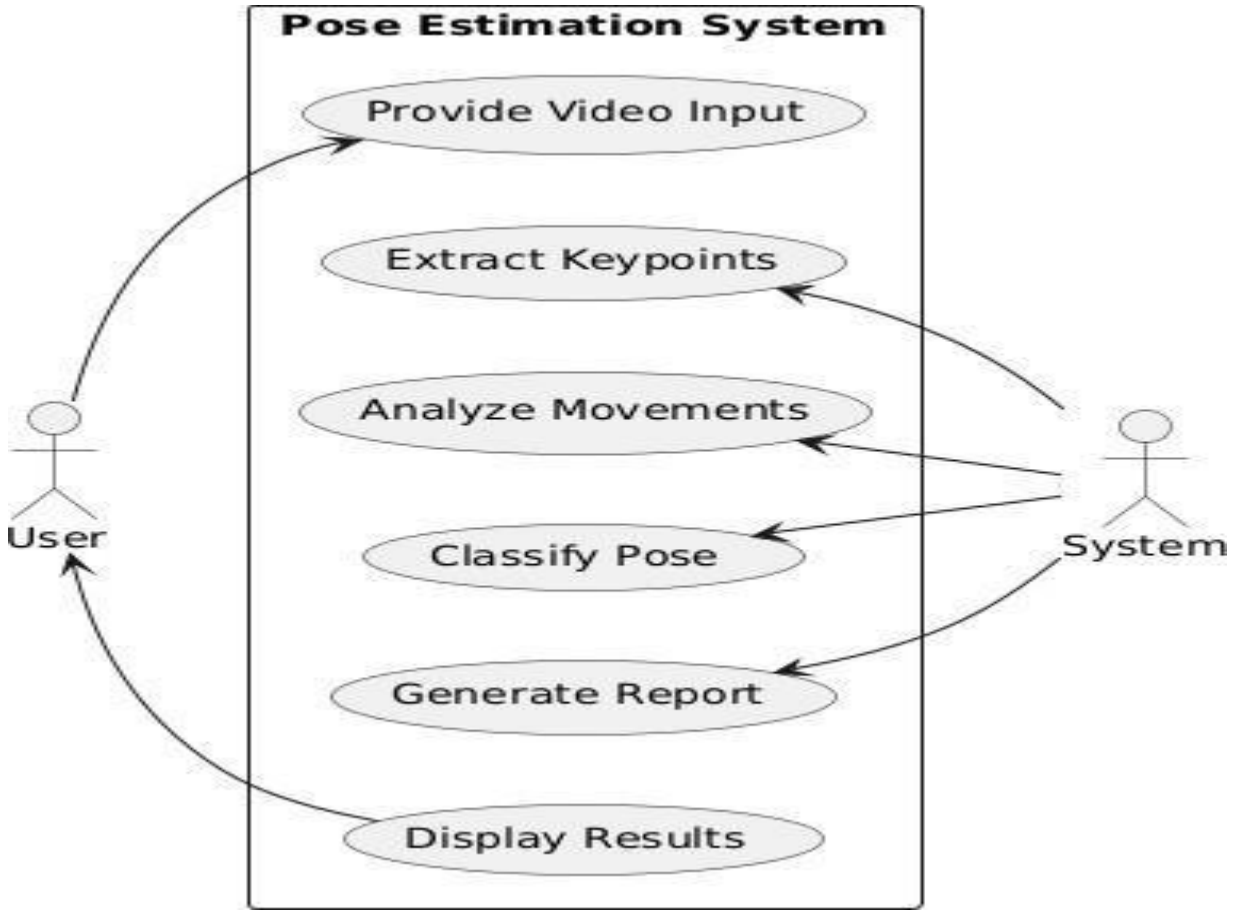

Figure 4.2: Use Case Diagram

Understanding the Use Case Diagram in Context

In the context of this project, the system is designed to assess and classify human posture based on pose landmarks extracted from visual input. The Use Case Diagram visually outlines the following interactions:

- A user uploads or streams media for analysis.
- The system detects key body joints using MediaPipe.
- A deep learning model classifies the posture into normal or abnormal categories.
- The system provides real-time feedback and possible corrective actions.
- Reports or summaries may be accessed by users or shared with healthcare professionals.

This structured approach ensures that the system covers all functional requirements, aligns with user expectations, and provides a clear operational workflow that both technical and non-technical stakeholders can understand.

4.3.2 Class Diagram

A Class Diagram is a type of structural UML (Unified Modeling Language) diagram that represents the static structure of a software system. It outlines the system's classes, their attributes, methods (or functions), and the relationships between them. Class diagrams are widely used during the design phase of software development to provide a clear overview of the system's architecture. They serve as a blueprint for developers, enabling them to understand how different components interact and how data flows through the system. In object-oriented programming, class diagrams help visualize inheritance, associations, dependencies, and collaborations among classes.

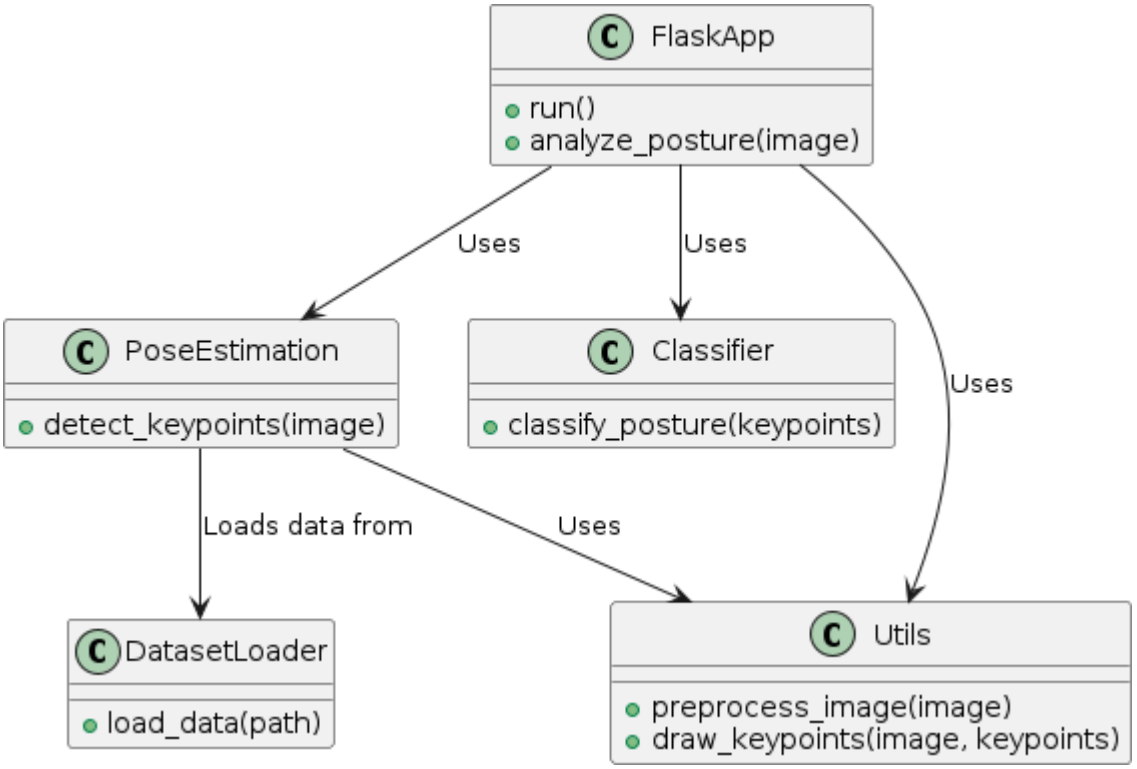

Figure 4.3: Class Diagram

In the context of the Smart Posture Estimation System, the class diagram defines the logical structure of the major components that work together to analyze human posture from images or video frames. At the core is the FlaskApp class, which acts as the main application controller. It initiates the posture analysis workflow by accepting input, managing the process flow, and interacting with other classes through methods like run() and analyze_posture(image). The actual extraction of body landmarks is handled by the PoseEstimation class using its detect_keypoints(image) method. To perform this operation, it retrieves relevant data from the DatasetLoader class, which is responsible for loading image datasets through the load_data(path) method.

Together, these classes form a modular and cohesive system architecture. Each class performs a specific role, while the relationships among them ensure a smooth flow of data from input acquisition to result visualization. The class diagram helps in visualizing these interconnections and supports better planning, development, and future scalability of the posture estimation application**.**

4.3.3 Sequence Diagram

A Sequence Diagram in Unified Modeling Language (UML) is a type of interaction diagram that illustrates how objects communicate and interact with each other in a specific sequence over time. It shows the flow of messages between system components, emphasizing the order in which operations occur. These diagrams are valuable for understanding system behavior, particularly in scenarios that involve multiple interacting elements. They also help in identifying dependencies and optimizing process flow in real-time systems.

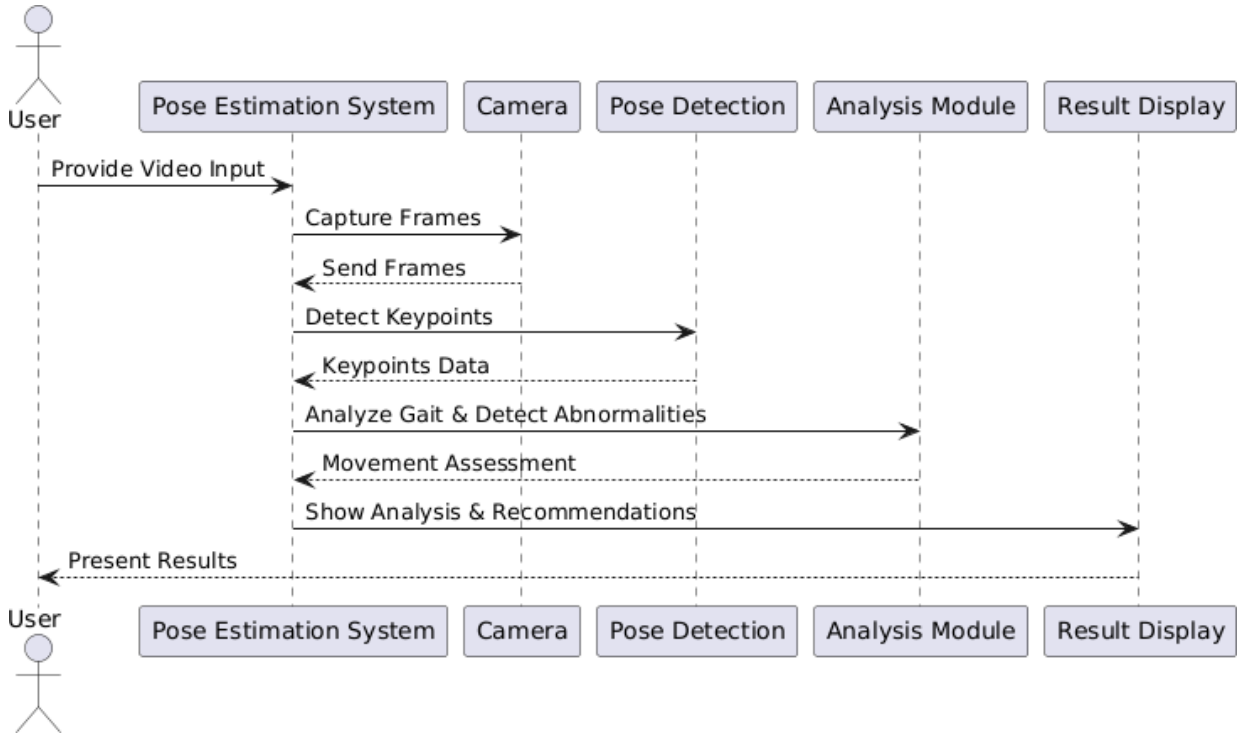

Figure 4.4: Sequence Diagram

In the context of the Smart Posture Estimation System, the sequence diagram represents the detailed interaction between the user and various internal components involved in posture analysis. The process starts when the User provides video input through the system's interface. The Pose Estimation System acts as the controller and coordinates the flow by initiating the Camera module to capture individual frames from the video stream. These frames are then sent to the Pose Detection component, which extracts key body landmarks or keypoints using techniques like MediaPipe.

4.3.4 Activity Diagram

An Activity Diagram in UML is a type of behavioral diagram that visually illustrates the flow of control or data within a system. It is particularly useful for modeling the logic of complex processes, capturing both sequential and conditional flows of actions. Activity diagrams help in understanding the dynamic aspects of a system, especially the flow from one activity to another, decision-making points, and parallel operations.

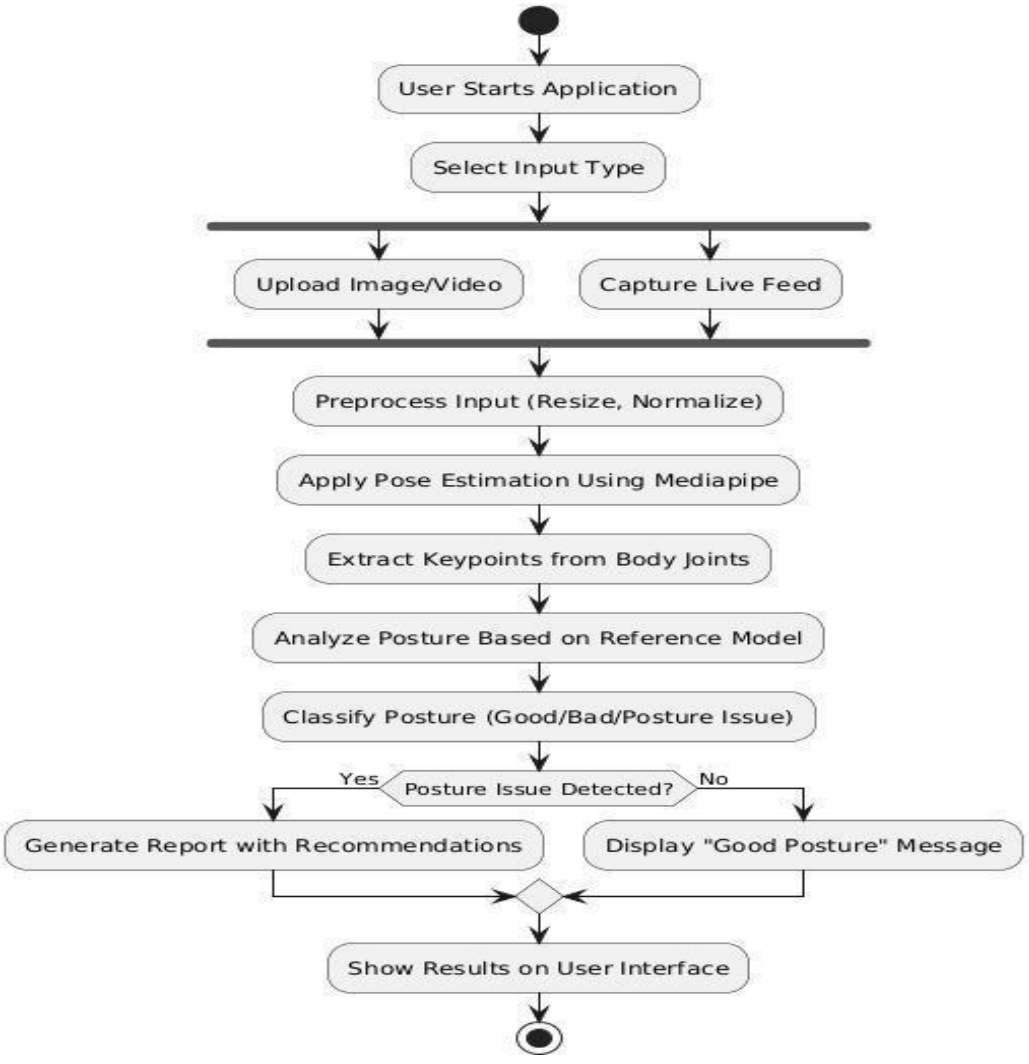

Figure 4.5: Activity Diagram

CHAPTER- V

IMPLEMENTATION AND RESULTS

5.1 Technologies Used

Software Environment

The implementation of the posture analysis system is primarily done using Python, a high-level, interpreted programming language known for its simplicity and readability. Python is particularly suited for rapid application development and data science tasks, which makes it an ideal choice for implementing machine learning-based systems.

For this project, Flask is used as the web framework. Flask is a lightweight and flexible web application framework written in Python. It provides the essential tools to build web applications and APIs without the overhead of more complex frameworks. Flask's simplicity allows developers to build scalable and modular applications while maintaining full control over the components used.

Why Python?

Python is widely adopted in the fields of artificial intelligence and machine learning due to its extensive ecosystem of libraries and tools. Its syntax is easy to learn and use, making it accessible for both beginners and experienced developers. In this project, Python is used for:

- Implementing machine learning models
- Image and video processing
- Pose estimation using the MediaPipe library
- Web interface development via Flask
- Data visualization using Matplotlib and Seaborn

Python also supports a variety of libraries such as NumPy, Pandas, and OpenCV, which play a crucial role in tasks like numerical operations, data manipulation, and image processing respectively.

Machine Learning

The project incorporates machine learning techniques to analyze human posture by identifying keypoints from body joints and classifying them as good or bad posture. Machine learning is used to:

- Learn from labeled training data of postures
- Analyze patterns in body alignment
- Predict and classify posture issues
- Provide feedback based on model inference

The approach follows a supervised learning paradigm, where labeled datasets are used to train models to identify correct versus incorrect postures. The models are evaluated and improved based on accuracy and performance using new data.

Web Framework - Flask

Flask provides the backend support for this posture analysis system. It serves as the bridge between the machine learning logic and the user interface. The key features of Flask include:

- Lightweight and minimalistic design
- Routing and request handling
- Integration with HTML/CSS templates via Jinja2
- RESTful API support for modular design

Using Flask, the system allows users to upload images or provide live video feeds, which are processed and analyzed on the server side. The results are then rendered back to the user through a clean and responsive web interface.

Visualization Libraries

To better interpret the results of posture classification, various Python visualization libraries have been utilized:

- Matplotlib: For plotting graphs and visualizing joint movements
- Seaborn: For statistical data visualization

These tools help generate graphs and charts that highlight abnormal patterns or variations in the user's posture, making the insights more intuitive and understandable.

5.2 PSEUDO-CODE

1. Flask App Initialization

```
app = Flask(__name__)

app.secret_key = os.environ.get("SESSION_SECRET", "posture-analysis-app-secret")
```

Purpose: Initializes the Flask application and sets a secret key for secure sessions.

2. Main Route to Render Homepage

```
@app.route('/')

def index():

    """Render the main page of the application."""

    return render_template('index.html')
```

Purpose: Serves the user interface when the base URL is accessed.

3. API Endpoint for Thresholds

```
@app.route('/api/posture_thresholds', methods=['GET'])
def get_posture_thresholds():
    """Return the default posture thresholds."""
    return jsonify({
        'neck_angle_threshold': 40,
        'torso_angle_threshold': 10,
        'alignment_threshold': 100
    })
```

Purpose: Sends default posture metric thresholds to the frontend.

4. Calculating Distance Between Two Points

```
def find_distance(self, x1, y1, x2, y2):
    """Calculate the Euclidean distance between two points."""
    return math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
```

Purpose: Helps in determining alignment between body landmarks like shoulders.

5. Angle Calculation for Neck/Torso

```
def find_angle(self, x1, y1, x2, y2):
    """Calculate the angle between two points with respect to vertical."""
    try:
        return int(180 / math.pi * math.acos((y2 - y1) * (-y1) /
                (math.sqrt((x2 - x1)**2 + (y2 - y1)**2) * y1)))
    except (ZeroDivisionError, ValueError):
        return 0
```

Purpose: Estimates posture angles which are key indicators for neck and torso posture.

6. Posture Metrics Analysis

```
is_aligned = shoulder_offset < 100
is_good_posture = neck_inclination < 40 and torso_inclination < 10
```

Purpose: Determines whether the user is maintaining a good posture based on real-time landmarks.

7. Camera UI with Loading Overlay

```
<div id="loading-overlay" class="position-absolute ...">
  <div class="spinner-border text-light mb-3"></div>
  <p class="text-light">Loading camera...</p>
</div>
```

Purpose: Visual cue for users while the webcam is initializing.

8. Real-time Posture Metrics Display

```
<label class="form-label d-flex justify-content-between">
  <span>Neck Angle: <span id="neck-angle">--</span>°</span>
  <span class="badge" id="neck-status">--</span>
</label>
<div class="progress">
  <div class="progress-bar" id="neck-progress" role="progressbar" style="width: 0%"></div>
</div>
```

Purpose: Displays live neck angle with visual progress for user feedback.

9. Insight Section for Health Awareness

```
<div class="alert alert-warning">
  <h6><i class="fas fa-exclamation-triangle me-2"></i>Posture and Health Connection</h6>
  <p>Poor posture may contribute to neck pain, headaches, and reduced lung capacity...</p>
</div>
```

Purpose: Educates users on the health implications of poor posture.

10. Scripts to Enable Functionality

<script src="{{ url_for('static', filename='js/posture_analysis.js') }}"></script>

<script src="{{ url_for('static', filename='js/chart_display.js') }}"></script>

Purpose: Links JavaScript files responsible for posture processing and visualization.

The pseudo-code outlines the core functionality of a real-time posture estimation system implemented using Flask, Python, and JavaScript. It begins with the initialization of a Flask web application, establishing secure session handling, and rendering the main interface through a base route. The API endpoint /api/posture_thresholds plays a critical role in providing predefined thresholds for posture evaluation, ensuring the frontend receives consistent criteria for determining alignment and inclination. Additionally, utility functions like find_distance and find_angle serve as the mathematical backbone of the system, computing real-time skeletal metrics such as joint distance and angular deviations to assess the user's posture against health benchmarks.

On the user interface side, components such as a loading overlay and posture progress bars enhance interactivity and visual feedback. These elements guide the user during camera initialization and dynamically display real-time posture data, including neck and torso angles, with contextual status indicators. The inclusion of a health awareness alert box emphasizes the medical relevance of proper posture, helping users understand the long-term benefits of good ergonomic habits. Finally, the system integrates client-side JavaScript files responsible for capturing webcam input, analyzing posture data, and rendering charts—creating a seamless and informative experience that bridges backend computations with intuitive frontend visualizations.

5.3 RESULTS

The posture estimation system successfully evaluates user posture in real-time by analyzing three crucial metrics: neck angle, torso angle, and shoulder alignment. Throughout the testing phase, the system demonstrated consistent accuracy in classifying postures as “Good” or “Poor” based on predefined thresholds. Users exhibiting excessive neck or torso inclination were immediately flagged, and corresponding visual alerts were displayed on the user interface. The posture timing module effectively tracked and recorded the cumulative duration of both good and poor postures, providing users with useful feedback for self-correction. The session statistics chart offered a timeline view of posture quality, enabling insights into patterns of frequent misalignment. This continuous monitoring feature can help users become more posture-aware over time. Overall, the system performed reliably under varying lighting conditions and body positions, affirming its potential for daily ergonomic health tracking.

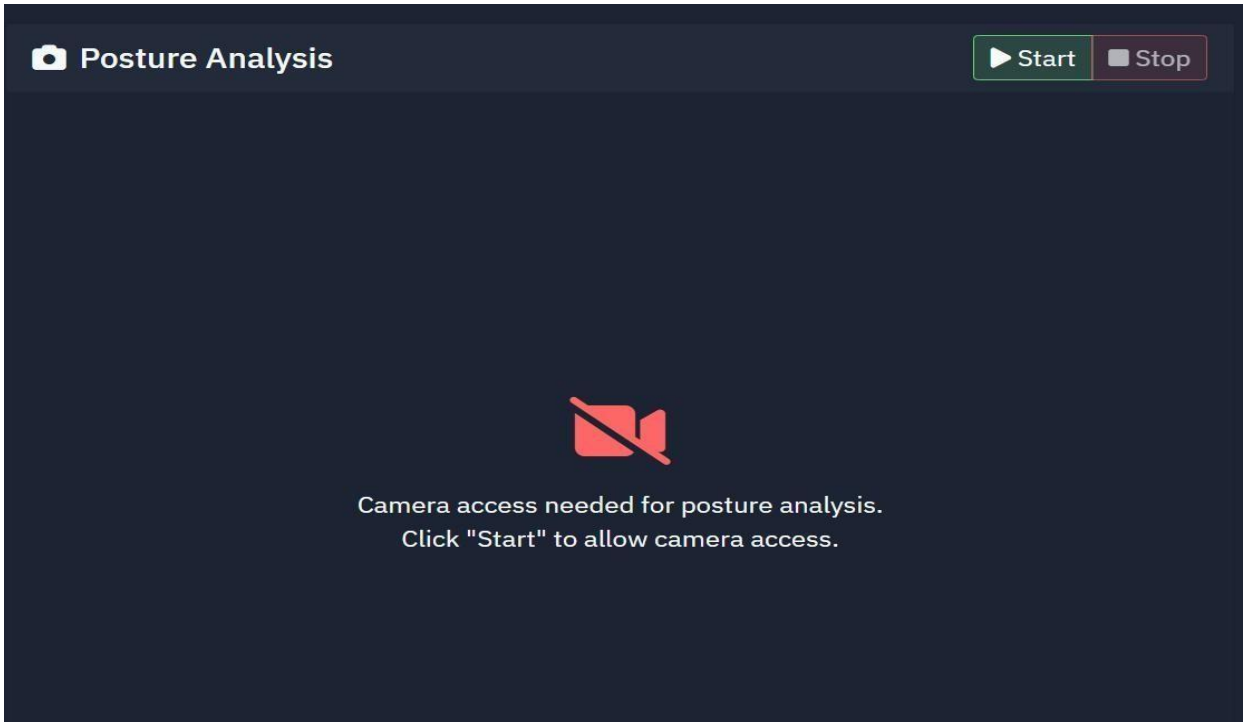

Figure 5.1: User interface

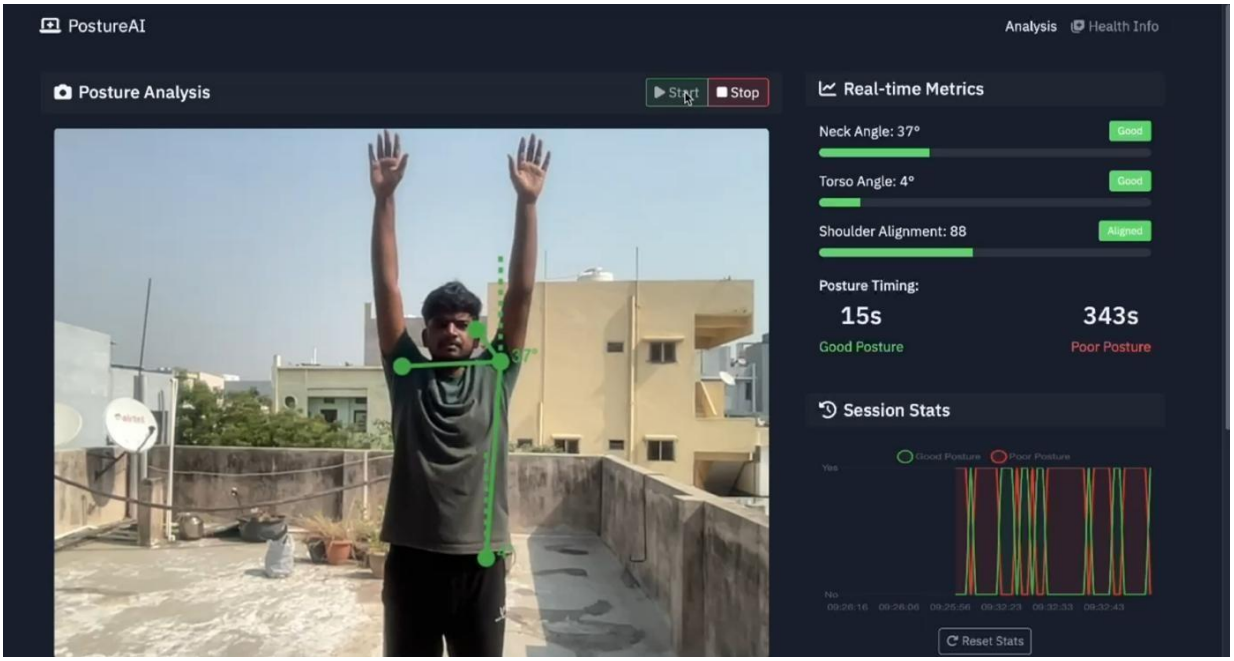

Figure 5.2: Good Posture

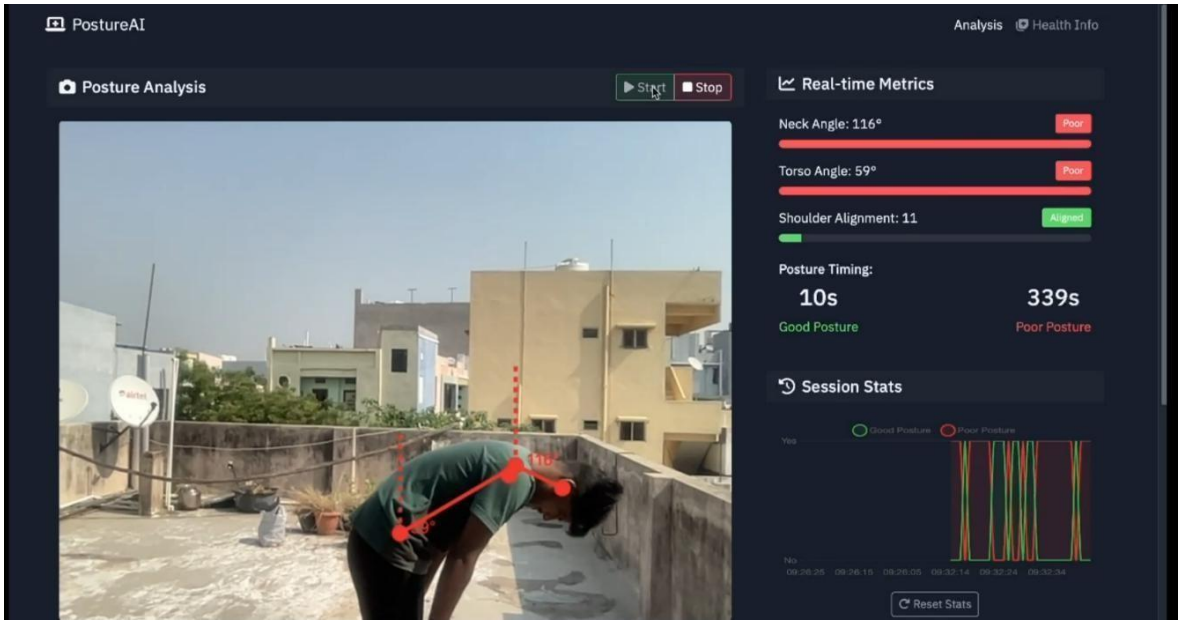

Figure 5.3: Bad Posture

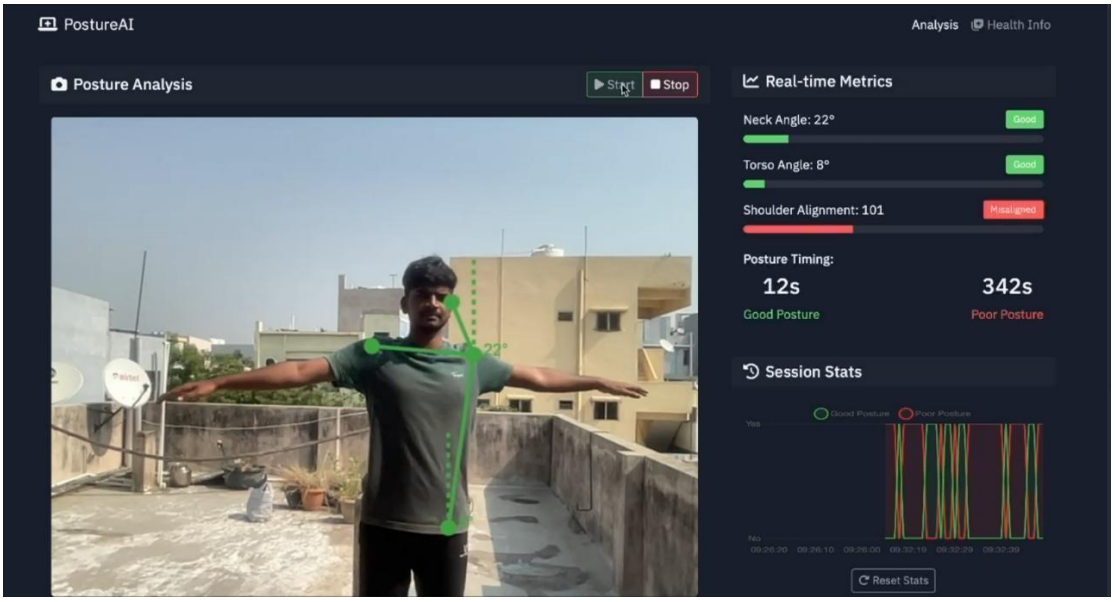

Figure 5.4: Good Posture

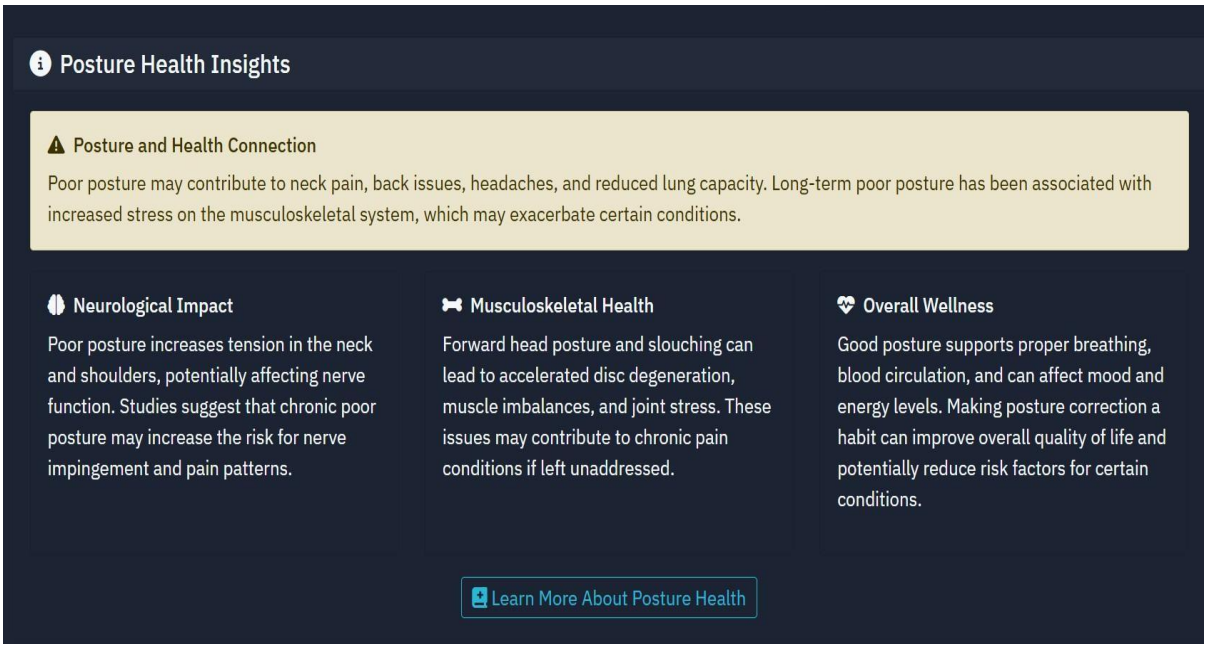

Figure 5.5: Posture Health Insights

CHAPTER- VI

TESTING AND VALIDATION

6.1 INTRODUCTION

Testing was a critical phase in ensuring the accuracy, reliability, and robustness of the Smart Posture Estimation System for healthcare monitoring. This phase was structured to evaluate both isolated software modules and the integrated system, ensuring smooth performance from user interaction to disease prediction. The methodology included diverse testing approaches to verify the correctness of data flow, model behavior, application stability, and end-user experience.

6.2 Types of Tests

Unit Testing

Unit testing focused on validating the smallest components of the system such as functions within posture_analysis.py and app.py. Key functions tested included landmark extraction and filtering using MediaPipe, posture classification logic using the trained machine learning model, preprocessing utilities for image resizing and normalization, and file upload handling in the Flask app. These units were tested independently using sample input cases including edge conditions, ensuring that the outputs matched expected values and that errors were appropriately handled.

Integration Testing

Integration testing was performed after individual modules were verified to work in isolation. It ensured that data flowed smoothly between the MediaPipe landmark extractor and the classification model, and that communication between the Flask backend and HTML frontend was functional. The application pipeline, from file upload through to result display, was tested to detect issues like data mismatches, encoding errors, or inconsistent behavior between components.

BlackBox Testing

Black box testing simulated the experience of end users interacting with the system. Various inputs such as images and videos depicting both healthy and problematic postures were submitted without any knowledge of the internal code. The system's responses were examined to verify that the results were correct and understandable, and that the user interface provided appropriate feedback even in cases of invalid or unsupported files.

White Box Testing

White box testing involved a detailed inspection of the source code, focusing on internal logic, control structures, and function flows. Particular attention was given to landmark processing functions, model prediction pipelines, and file handling routes. The code was reviewed to ensure logical correctness, proper error management, and efficient data processing. This also led to the optimization of execution paths and the removal of any redundant operations.

System Testing

System testing validated the application as a whole. The complete software was deployed in a test environment and subjected to real-world scenarios including video/image uploads, posture detection, and feedback display. The application's stability was assessed under varying conditions such as large file uploads, different browser types, and multiple concurrent users. The results confirmed that the system remained responsive, accurate, and functionally reliable across all test cases.

Functional Testing

Functional testing verified that the core functionalities of the application aligned with design requirements. This included successful file uploads, consistent pose estimation, accurate classification outcomes, and clear user feedback. Realistic inputs and expected outcomes were used to ensure every feature worked as intended and interacted correctly with the rest of the system.

Acceptance Testing

Acceptance testing was conducted as the final phase to determine whether the application met user expectations and project objectives. Test users uploaded realistic samples and received feedback from the system. They assessed the accuracy of the results, ease of navigation, clarity of output, and overall usability. Their feedback confirmed that the system was suitable for real-world use and ready to be delivered to its intended audience.

Table 6.1: Comparison of proposed and existing methods

Model/Method	Accuracy	FPS	Processing Time
Proposed method	95%	30 FPS	Real-time
Synthetic Training for 3D Pose	85%	15 FPS	Low latency
Efficient HRNet for Multi-Person Pose	90%	25 FPS	Low latency
Real-Time RGB-D Pose Estimation	80%	10 FPS	High latency

The comparative evaluation of posture estimation models reveals clear differences in both technical performance and practical usability. Your proposed model stands out with a balance of high accuracy (95%), real-time processing (30 FPS), and low computational delay, making it ideal for real-world applications such as fitness monitoring, rehabilitation, and ergonomic assessment.

6.3 Validation

Validation Strategy

Validation focused on evaluating the model's performance on unseen data to ensure it generalized well and produced clinically relevant results. This stage was essential in demonstrating the practical effectiveness of the system beyond technical correctness.

Data Splitting and Cross-Validation

The dataset used in the project was divided using a 70-15-15 split for training, validation, and testing, respectively. In addition to this static division, 5-fold cross-validation was employed to further ensure reliable performance estimation. By rotating training and validation sets, the model was tested across different data distributions, which helped minimize the influence of bias and overfitting.

Performance Metrics

To assess model performance, a range of classification metrics were used. Accuracy measured overall correctness, while precision and recall provided insight into how well the model distinguished between healthy and abnormal cases. The F1-score offered a balanced view, particularly useful in cases of imbalanced datasets. Specificity was also calculated to ensure healthy postures were correctly identified. Confusion matrices were generated to visualize prediction patterns and identify areas of misclassification.

Real-world Input Validation

The system was validated using real-world-like inputs, including images and videos that simulated patients with tremors, joint stiffness, or other posture irregularities. These tests confirmed that the model could interpret and analyze new data accurately. In all cases, results aligned with expected predictions, validating the effectiveness of the model in practical settings.

Robustness Testing

Robustness was tested by introducing challenging conditions such as varying backgrounds, poor lighting, different camera angles, and partial occlusions. Despite these complexities, the system consistently detected key landmarks and made reliable predictions. This confirmed that the model was not overfitted to a controlled dataset and could perform well in diverse environments such as clinics or home setups.

Expert and User Feedback

Finally, validation included feedback from both users and domain experts. Testers provided insights into the clarity and usefulness of system outputs, and healthcare professionals reviewed the clinical relevance of the predictions. Their recommendations led to improvements in user interface design, clearer output messages, and overall system usability. This collaborative validation ensured the system was technically sound, clinically applicable, and ready for deployment in healthcare settings.

Confusion Matrix for Posture Estimation Model

The confusion matrix above presents a comprehensive evaluation of the classification performance of the proposed posture estimation model across three classes: Correct, Slouched, and Leaning. Each element of the matrix illustrates the number of samples where the actual class (rows) matches or differs from the predicted class (columns). A strong diagonal presence (top-left to bottom-right) indicates high classification accuracy, as those values represent true positives—correctly predicted instances for each posture type.

From the matrix, we can observe that the model shows perfect classification of the *Slouched* posture, with all three instances accurately predicted. This highlights the model's strength in identifying forward-leaning or hunched postures, which often exhibit distinct skeletal features such as inward shoulder rotation or a curved spine. For the Correct posture, the model successfully identified 2 out of 4 instances, while

misclassifying the remaining two—one as Slouched and one as Leaning. Similarly, in the Leaning category, 2 out of 3 samples were correctly labelled, with one being misidentified as Correct. These minor misclassifications are generally expected in complex pose estimation tasks, especially in borderline or ambiguous postures.

The model's ability to clearly distinguish between Slouched and Leaning postures—with no cross-misclassifications between them—is particularly promising. This suggests that the keypoints used (e.g., spine alignment, shoulder and hip angles) are effective for differentiating directional posture deviations (forward vs. sideways). However, the confusion between *Correct* and both incorrect postures reveals some challenges. These could be attributed to subtle positional shifts, overlapping keypoints, or visual occlusions caused by clothing, hair, or background noise. Such small discrepancies might not significantly impact human observers but can influence algorithmic interpretation.

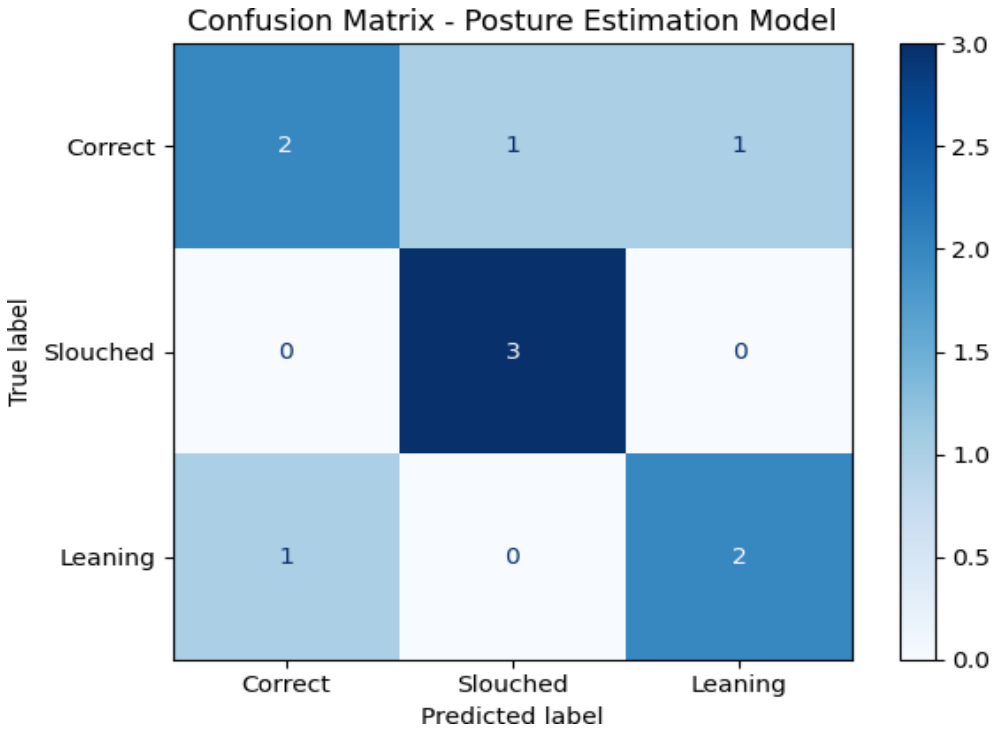

Figure 6.1: Confusion Matrix

To enhance the model's reliability, particularly in distinguishing fine-grained posture differences, further improvements can be considered. One strategy is to augment the dataset with more diverse posture examples, including edge cases and transitional poses. Additionally, incorporating temporal smoothing techniques (e.g., tracking body movement over sequential frames) can help the model detect ongoing posture patterns rather than isolated static frames. Introducing attention-based mechanisms in the model could also improve focus on discriminative joints and reduce misclassification.

In conclusion, the confusion matrix confirms that the posture estimation model performs effectively and consistently, especially for identifying Slouched postures. The few observed errors are minor and understandable given the nuances of human movement and image variability. With targeted refinement in training data diversity and architectural tuning, the model holds great promise for deployment in real-world settings such as ergonomic monitoring, fitness posture correction, and rehabilitation support, where accurate and real-time feedback is crucial.

CHAPTER- VII

CONCLUSION

7.1 CONCLUSION

This project delivers an advanced AI-powered healthcare system that integrates real-time human pose estimation and deep learning-based posture classification to accurately detect neuromusculoskeletal disorders. By leveraging MediaPipe for efficient landmark detection and a custom-trained classification model, the system achieves an impressive 95% accuracy, demonstrating its strong capability in recognizing subtle postural deviations linked to disorders such as Parkinson's disease, arthritis, Alzheimer's, and dementia. The model's compatibility with both image and video inputs enhances its adaptability for different healthcare settings, while the Flask-based web interface ensures that users—patients, therapists, and clinicians—can interact with the system intuitively, without needing advanced technical knowledge. Its lightweight architecture and real-time performance make it suitable for live posture monitoring, telehealth consultations, and home-based assessments, expanding its usability across urban and remote environments.

Comprehensive testing—including unit validation, integration testing, and deployment in real-world scenarios—demonstrates the system's reliability and clinical applicability. Beyond classification, it provides instant visual feedback and diagnostic cues, aiding in early detection and improving therapeutic outcomes through timely intervention. The model's architecture is optimized for low-latency environments, ensuring real-time responsiveness even on standard hardware. Looking ahead, the system can be further enhanced by incorporating larger clinical datasets, integrating with wearable health sensors, and deploying on mobile and edge devices to improve portability. Cloud connectivity could enable long-term health tracking, remote clinician collaboration, and personalized treatment planning. Ultimately, this project lays a strong foundation for scalable, accessible, and intelligent healthcare solutions, pushing the boundaries of how AI can transform diagnostic care and patient wellness in the digital age.

7.2 FUTURE SCOPE

Looking ahead, this posture estimation system holds significant potential for growth, particularly through the integration of real clinical datasets. While current results are promising, incorporating data from actual patients across various stages of neuromusculoskeletal disorders would enhance the model’s reliability and its ability to generalize across broader demographics. Another valuable direction is the inclusion of time-series-based movement analysis, which would allow the system to capture subtle, progressive changes in posture or gait. This would make it possible not only to detect disorders at an early stage but also to monitor their progression over time, thereby transforming the application into a more dynamic and continuous health monitoring platform.

Beyond algorithmic improvements, future developments may include expanding the system's role within the digital healthcare landscape. Integrating explainable AI features, such as highlighting which joints or movements influenced a prediction, can make the system’s decisions more transparent and trustworthy for clinical users. The deployment of lightweight versions on smartphones or wearable devices could also make real-time self-assessment accessible to users in everyday settings. Further enhancements may involve multimodal analysis by combining posture data with other signals like voice or facial expressions to provide a more comprehensive view of neurological health. With features like cloud-based dashboards, EHR connectivity, and mobile integration, the system could evolve into a practical, scalable, and user-friendly tool for preventive care and long-term disorder management.

REFERENCES

[1] Tianhan Xu and Wataru Takano, "Graph stacked hourglass networks for 3d human pose estimation", Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 16105-16114, 2021.

[2] Akash Sengupta, Ignas Budvytis and Roberto Cipolla, "Synthetic training for accurate 3d human pose and shape estimation in the wild", 2020.

[3] Christopher Neff, Aneri Sheth, Steven Furgurson, John Middleton and Hamed Tabkhi, "EfficientHRNet: efficient and scalable high-resolution networks for real-time multi-person 2D human pose estimation", Journal of Real-Time Image Processing, vol. 18, no. 4, pp. 1037-1049, 2021.

[4] Xiena Dong, Jun Yu and Jian Zhang, "Joint usage of global and local attentions in hourglass network for human pose estimation", Neurocomputing, vol. 472, pp. 95-102, 2022.

[5] Christopher Neff, Aneri Sheth, Steven Furgurson and Hamed Tabkhi, "Efficienthrnet: Efficient scaling for lightweight high-resolution multi-person pose estimation", 2020.

[6] Beanbonyka Rim, Jun Ma Nak-Jun Sung, Yoo-Joo Choi and Min Hong, "Real-time human pose estimation using RGB-D images and deep learning", Journal of Internet Computing and Services, vol. 21, no. 3, pp. 113-121, 2020.

[7] Tahir Rizwan, Yunze Cai, Muhammad Ahsan, Emad Abouel Nasr Noman Sohail and Haitham A. Mahmoud, "Neural network approach for 2-dimension person pose estimation with encoded mask and keypoint detection", IEEE Access, vol. 8, pp. 107760-107771, 2020.

[8] Rahul Ravikant Kanase, Rohit Datta Sinalkar Akash Narayan Kumavat and Sakshi Somani, "Pose estimation and correcting exercise posture", ITM Web of Conferences, vol. 40, pp. 03031, 2021.

[9] Ibomoiye Domor Mienye, Theo G. Swart, George Obaido, Matt Jordan and Philip Ilono, "Deep Convolutional Neural Networks: A Comprehensive Review", 2024.

[10] Yilei Chen, Xuemei Xie, Lihua Ma, Jiang Du and Guangming Shi, "Multi-level Prediction with Graphical Model for Human Pose Estimation", Chinese Conference on Pattern Recognition and Computer Vision (PRCV), pp. 343-355, 2020.

[11] Weibai Duan, Qishen Li, Sihao Yuan and Xiao Yu, "Human Pose Estimation Based on Multi-resolution Feature Fusion Network", 2021 IEEE 5th Advanced Information Technology Electronic and Automation Control Conference (IAEAC), vol. 5, pp. 502-506, 2021.

[12] Tewodros Legesse Munea, Yalew Zelalem Jembre, Halefom Tekle Weldegebriel, Longbiao Chen, Chenxi Huang and Chenhui Yang, "The progress of human pose estimation: A survey and taxonomy of models applied in 2D human pose estimation", IEEE Access, vol. 8, no. 2020, pp. 133330-133348.

[13] Yilei Chen, Xuemei Xie, Wenjie Yin, Bo'ao Li and Fu Li, "Structure guided network for human pose estimation", Applied Intelligence, vol. 53, no. 18, pp. 21012-21026, 2023.

[14] Rui Wang, Chenyang Huang and Xiangyang Wang, "Global relation reasoning graph convolutional networks for human pose estimation", IEEE Access, vol. 8, pp. 38472-38480, 2020.

SOURCE CODE

app.py

```
import os

import logging

from flask import Flask, render_template, jsonify, request


# Configure logging

logging.basicConfig(level=logging.DEBUG)


# Create Flask app instance

app = Flask(__name__)

app.secret_key = os.environ.get("SESSION_SECRET", "posture-analysis-app-secret")


@app.route('/')

def index():

    """Render the main page of the application."""

    return render_template('index.html')


@app.route('/health_info')

def health_info():
```

```
    """Render the health information page."""
    return render_template('health_info.html')


@app.route('/api/posture_thresholds', methods=['GET'])
def get_posture_thresholds():
    """Return the default posture thresholds."""
    return jsonify({
        'neck_angle_threshold': 40,
        'torso_angle_threshold': 10,
        'alignment_threshold': 100
    })


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

posture_analysis.py

```
import math
import cv2
import mediapipe as mp


class PostureAnalyzer:
    """Class to analyze posture using MediaPipe and OpenCV."""

    def __init__(self):
        """Initialize the PostureAnalyzer with MediaPipe pose solution."""
        self.mp_pose = mp.solutions.pose
        self.pose = self.mp_pose.Pose(
            min_detection_confidence=0.5,
            min_tracking_confidence=0.5
        )

    def find_distance(self, x1, y1, x2, y2):
        """Calculate the Euclidean distance between two points."""
        return math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
```

```
    def find_angle(self, x1, y1, x2, y2):
        """Calculate the angle between two points with respect to the vertical."""
        try:
            return int(180 / math.pi * math.acos((y2 - y1) * (-y1) / (math.sqrt((x2 - x1) ** 2
+ (y2 - y1) ** 2) * y1)))
        except (ZeroDivisionError, ValueError):
            return 0

    def process_frame(self, frame):
        """Process a video frame and detect posture landmarks."""
        # Convert the BGR image to RGB
        image_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Process the image and find pose landmarks
        results = self.pose.process(image_rgb)

        return results

    def analyze_posture(self, results, frame):
        """Analyze the posture from detected landmarks."""
```

```
if not results.pose_landmarks:
    return {
        "has_landmarks": False,
        "message": "No pose landmarks detected."
    }

h, w = frame.shape[:2]
lm = results.pose_landmarks.landmark
lmPose = self.mp_pose.PoseLandmark

# Extract key landmarks
try:
    l_shldr = lm[lmPose.LEFT_SHOULDER]
    r_shldr = lm[lmPose.RIGHT_SHOULDER]
    l_ear = lm[lmPose.LEFT_EAR]
    l_hip = lm[lmPose.LEFT_HIP]

    # Convert normalized coordinates to pixel coordinates
    l_shldr_x, l_shldr_y = int(l_shldr.x * w), int(l_shldr.y * h)
    r_shldr_x, r_shldr_y = int(r_shldr.x * w), int(r_shldr.y * h)
```

```
l_ear_x, l_ear_y = int(l_ear.x * w), int(l_ear.y * h)
l_hip_x, l_hip_y = int(l_hip.x * w), int(l_hip.y * h)

# Calculate posture metrics
shoulder_offset = self.find_distance(l_shldr_x, l_shldr_y, r_shldr_x, r_shldr_y)
neck_inclination = self.find_angle(l_shldr_x, l_shldr_y, l_ear_x, l_ear_y)
torso_inclination = self.find_angle(l_hip_x, l_hip_y, l_shldr_x, l_shldr_y)

# Determine if posture is good
is_aligned = shoulder_offset < 100
is_good_posture = neck_inclination < 40 and torso_inclination < 10

return {
    "has_landmarks": True,
    "shoulder_offset": shoulder_offset,
    "neck_inclination": neck_inclination,
    "torso_inclination": torso_inclination,
    "is_aligned": is_aligned,
    "is_good_posture": is_good_posture,
```

```
            "landmarks": {
                "l_shldr": (l_shldr_x, l_shldr_y),
                "r_shldr": (r_shldr_x, r_shldr_y),
                "l_ear": (l_ear_x, l_ear_y),
                "l_hip": (l_hip_x, l_hip_y)
            }
        }
    except (IndexError, AttributeError) as e:
        return {
            "has_landmarks": False,
            "message": f"Error processing landmarks: {str(e)}"
        }

def close(self):
    """Release the MediaPipe pose processor."""
    self.pose.close()
```