Parkincurator: Multi-Modal Approach for Parkinson's Disease Detection, Classification, and Management

Submitted by:

Ishan Agrawal (20EBKAI011) and Ira Goyal (20EBKCS050)

Prof. Himanshu Verma

DEPARTMENT OF COMPUTER SCIENCE

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHONOLOGY

IN

ARTIFICIAL INTELLIGENCE and COMPUTER SCIENCE

at

B.K. BIRLA INSTITUTE OF ENGINEERING & TECHNOLOGY

PILANI, RAJASTHAN (INDIA) - 333031

(AFFILIATED TO BIKANER TECHNICAL UNIVERSITY, BIKANER, RAJASTHAN (INDIA))

JUNE, 2024

DECLARATION

I hereby declare that the project entitled “Parkincurator: Multi-Modal Approach for Parkinson’s Disease Detection, Classification, and Management” submitted for the B. Tech. (AI) and B. Tech. (CSE) degree is my original work and the project has not formed the basis for the award of any other degree, or any other similar title.

[illegible] Ira Goyal

Signature of the Student

(Ishan Agrawal & Ira Goyal)

Place: B K Birla Institute of Engineering and Technology, Pilani

Date: 07-06-2024

CERTIFICATE

This is to certify that the project titled "Parkincurator: Multi-Modal Approach for Parkinson's Disease Detection, Classification, and Management" is the bonafide work carried out by Ishan Agrawal (20EBKAI011) , a student of B Tech (AI) and Ira Goyal (20EBKCS050), a student of B Tech (CSE) of B. K. Birla Institute of Engineering & Technology, Pilani affiliated to Bikaner Technical University, Bikaner, Rajasthan(India) during the academic year 2020-24 in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Artificial Intelligence) & Bachelor of Technology (Computer Science and Engineering) and that the project has not formed the basis for the award previously of any other degree, or any other similar title.

Signature of the Guide

Place:

Date:

ABSTRACT

Parkinson's Disease (PD) poses a significant challenge in the domain of healthcare, its insidious progression marked by the relentless erosion of dopamine-producing neurons within the brain. This debilitating process gives rise to a cascade of motor impairments, including tremors, rigidity, and bradykinesia, which significantly diminish the quality of life for affected individuals. In response to this pressing challenge, this study embarks on a pioneering journey towards redefining the paradigm of PD management, integrating cutting-edge technologies with innovative methodologies to confront the complexities of this neurodegenerative condition. The overarching purpose of this research is to forge an unyielding framework capable of early detection and tailored management of PD. Central to this framework are three primary modalities: intricate handwriting analysis, voice input analysis, and sophisticated body movement analysis. Handwriting analysis serves as a window into the subtleties of motor function, revealing micrographia —a characteristic sign of PD. Voice input analysis detects tremors and other speech abnormalities indicative of the disease, while body movement analysis discerns bradykinesia and rigidity, crucial motor symptoms that define PD progression.

The methodology employed in this endeavour is both comprehensive and meticulous. Leveraging Convolutional Neural Networks (CNN) for handwriting analysis, XG Boost for voice input analysis, and Media Pipe for body movement analysis, the research team conducts exhaustive training on extensive datasets. This rigorous approach ensures not only the accuracy of disease stage classification but also enables the provision of personalized recommendations tailored to each patient's unique presentation. By tailoring interventions to individual needs, the proposed framework aims to optimize therapeutic outcomes and enhance patient well-being.

A noteworthy innovation of this study is the incorporation of a real-time fall detection mechanism—a critical addition aimed at mitigating the risks associated with PD-related falls. Engineered to swiftly alert caregivers in the event of a fall, this mechanism serves as a lifeline, offering reassurance to patients and their families while bolstering patient safety. The principal findings of this research underscore the remarkable efficacy of the proposed framework in PD detection and classification across diverse modalities. Furthermore, the real-time fall detection mechanism emerges as a transformative addition, symbolizing a pivotal advancement in patient care. In summation, this research represents a significant leap forward in PD management, harnessing the power of technology to address the multifaceted challenges posed by this complex neurological disorder. Through its holistic approach and commitment to personalized care, the proposed framework holds the promise of improving patient outcomes, slowing disease progression, and ultimately enhancing the quality of life for individuals living with Parkinson's Disease.

ACKOWLEGEMENT

I am both proud and deeply humbled to extend my sincere appreciation to the various individuals who have played instrumental roles in guiding and supporting me throughout the preparation of this report. Firstly, my heartfelt thanks go to our mentor Mr. Himanshu Verma whose unwavering support, encouragement, and the provision of invaluable videos and interface facilities have significantly contributed to the successful completion of this report.

I would also like to express my profound gratitude to my teachers and Heads of Department (HODs) within the department for their continuous support and encouragement during my internship. Their guidance has been pivotal in navigating the challenges and ensuring the successful culmination of this learning experience.

Last but certainly not least, I want to convey my deepest thanks to my parents, whose unwavering financial support has been the cornerstone of my education in this college and during my training. Their constant encouragement, motivation, and belief in my abilities have been a driving force in my pursuit of knowledge and continuous learning.

The collective efforts of these individuals have not only enriched my academic and professional journey but have also fostered an environment of growth, resilience, and determination. I am truly grateful for the privilege of having such a supportive network of mentors, instructors, and family members in my educational and professional endeavors.

LIST OF FIGURES

TABLE OF CONTENT

1. INTRODUCTION

1.1 PROBLEM DEFINITION

Parkinson's disease (PD) presents a complex array of symptoms that severely affect the quality of life of individuals afflicted by it. One of the hallmark manifestations of PD is the deterioration of motor function, encompassing a spectrum of issues ranging from tremors and bradykinesia to postural instability and gait disturbances. These motor symptoms often lead to debilitating consequences, including falls, diminished dexterity, and compromised mobility.

Among the distinct motor impairments associated with PD, handwriting difficulties, medically termed "micrographia," pose a significant challenge. Handwriting, being a fundamental aspect of communication and daily functioning, becomes progressively impaired as the disease advances. This deterioration not only affects the legibility of written communication but also reflects the underlying motor control deficits and fine motor skill impairments characteristic of PD. Furthermore, speech impairment, commonly referred to as "dysarthria," emerges as another debilitating aspect of PD. Individuals with PD often experience slurred speech, reduced volume, monotone voice, and articulation difficulties, which not only hinder effective communication but also contribute to social withdrawal and diminished quality of life.

Moreover, the manifestation of tremors, particularly in the hands and limbs, exacerbates the challenges faced by individuals with PD. These involuntary rhythmic oscillations not only impede fine motor tasks like writing but also interfere with activities of daily living, such as eating, dressing, and hygiene maintenance.

Additionally, PD encompasses non-motor symptoms that further compound the burden on individuals. These include cognitive impairment, mood disturbances, and autonomic dysfunction, all of which significantly impact overall well-being and functional independence.

Addressing the multifaceted challenges posed by PD requires a comprehensive understanding of the interplay between motor and non-motor symptoms, as well as their impact on daily functioning and quality of life. By elucidating the intricate mechanisms underlying these challenges, advancements in therapeutic interventions, management strategies, and supportive care can be developed to alleviate the burden experienced by individuals living with PD and enhance their overall quality of life.

1.2 PROJECT OVERVIEW/SPECIFICATIONS & ELABORATIONS

Parkinson's Disease (PD) is a chronic and progressive neurodegenerative disorder characterized by the loss of dopamine-producing neurons in the brain. This loss leads to a range of motor symptoms, including tremors, rigidity, bradykinesia, and postural instability, as well as non-motor symptoms such as cognitive impairment and autonomic dysfunction. The management of PD traditionally revolves around symptom alleviation through medication, physical therapy, and, in some cases, surgical interventions like deep brain stimulation. However, existing approaches often lack precision in early detection and personalized management, leading to suboptimal outcomes and challenges in maintaining patient well-being.

This project seeks to address these limitations by developing an integrated framework that harnesses the power of advanced technologies and methodologies to redefine PD management. By leveraging artificial intelligence (AI), machine learning (ML), and image processing technologies, the goal is to enable early detection, accurate classification, and personalized intervention tailored to the specific needs of each patient.

1.2.1. Objectives

- Develop a comprehensive framework for early detection and personalized management of Parkinson's disease.
- Utilize handwriting analysis, voice input analysis, and body movement analysis as primary modalities for PD detection and classification.
- Implement image processing advanced machine learning algorithms, including Convolutional Neural Networks (CNN) for handwriting analysis and XGBoost for voice input analysis.
- Provide personalized recommendations tailored to each patient's unique presentation to optimize therapeutic outcomes and enhance patient well-being.

1.2.2. Methodology

a. Handwriting Analysis:

- Utilize Convolutional Neural Networks (CNN) for feature extraction and classification of handwriting patterns.
- Train the CNN model on a large dataset of handwritten samples, including those from individuals with PD and healthy controls.
- Created a user friendly UI for image clicking and uploading to process and predict on hand writing images.
- Include different data like wave pattern and spiral pattern

b. Voice Input Analysis:

- Employ XGBoost algorithm for voice input analysis to detect tremors and speech abnormalities indicative of PD.
- Extract acoustic features such as pitch, intensity, and formant frequencies from voice recordings.
- Train the XGBoost model on a diverse dataset of voice samples collected from individuals with varying stages of PD and healthy controls.

c. Body Movement Analysis:

- Utilize MediaPipe framework for body movement analysis to discern bradykinesia and rigidity, key motor symptoms of PD.
- Extract skeletal pose features from video recordings of movement tasks using MediaPipe's pose estimation model.
- Train machine learning algorithms, such as Support Vector Machines (SVM) or Random Forests, to classify movement patterns associated with PD.
- Develop algorithms for real-time assessment of motor symptoms and disease progression.

d. Real-time Fall Detection Mechanism:

- Always runs in background to monitor patients
- Used image processing and KNN algorithm to detect the fall
- Utilize computer vision techniques to analyze video streams or webcam feeds for changes in posture indicative of a fall.
- Trigger immediate alerts to caregivers or emergency services in case of a fall.

1.2.3. Deliverables

- A fully developed framework for early detection and personalized management of PD.
- Software applications for handwriting analysis, voice input analysis, and body movement analysis.
- Integration of real-time fall detection mechanism into existing PD management systems.

1.2.4. Expected Impact

- Improved early detection and diagnosis of PD, leading to timely intervention and better outcomes for patients.
- Personalized care plans tailored to individual needs, optimizing therapeutic interventions and enhancing patient well-being.
- Enhanced safety and quality of life for individuals with PD through real-time monitoring and fall detection mechanisms.

1.3 HARDWARE SPECIFICATIONS

1.3.1 Memory (RAM):

- Recommendation: At least 8GB of RAM.
- Reasoning: Sufficient RAM ensures smooth operation of the software, especially when handling multimedia data from the camera and microphone.
- Benefits: With ample RAM, the software can efficiently process and manipulate multimedia data without experiencing slowdowns or crashes.

1.3.2 Processor (CPU):

- Recommendation: A modern processor with multiple cores, such as Intel Core i5 or higher, or AMD Ryzen 5 or higher.
- Reasoning: A powerful CPU is crucial, particularly when dealing with tasks like processing video or audio streams.
- Benefits: A capable CPU ensures smooth operation and responsiveness of the software, even during resource-intensive tasks.

1.3.3 Storage:

- Recommendation: 256GB solid-state drive (SSD).
- Reasoning: Sufficient storage space is required for the operating system and any multimedia files that will be captured or processed.
- Benefits: SSDs provide faster data access compared to traditional hard disk drives (HDDs), resulting in quicker software loading times and smoother overall performance.

1.3.4 Camera:

- Recommendation: Any standard webcam.
- Reasoning: A webcam allows users to capture video for fall detection monitoring and Parkinson stage prediction also it allows user to click and check handwriting based Parkinson prediction.

1.3.5 Microphone:

- Recommendation: Any standard microphone, such as those built into laptops or external USB microphones.
- Reasoning: A microphone enables users and capture and predict Parkinson disease using audio.

1.3.6 Operating System:

- Compatibility: Windows 7+ (64 Bit), macOS, and Linux.
- Reasoning: The software is compatible with multiple operating systems, providing flexibility for users with different preferences or requirements.
- Benefits: Users can choose their preferred operating system without compatibility issues, allowing them to work with the software on their preferred platform.

1.4 Software Specification for Parkinson's Disease Management Framework

- **VS CODE**

Visual Studio Code (VS Code) is an excellent choice for writing Python code due to its versatility, ease of use, and robust features tailored for Python development. With its lightweight yet powerful design, VS Code provides a seamless coding experience for Python developers.

Firstly, VS Code offers comprehensive support for Python syntax highlighting, code completion, and automatic indentation, enhancing code readability and productivity. Its IntelliSense feature provides intelligent code suggestions, helping developers write Python code more efficiently and accurately.

Additionally, VS Code integrates seamlessly with popular Python frameworks and libraries such as Django, Flask, and NumPy, enabling developers to build and manage Python projects with ease. The built-in terminal allows for running Python scripts directly within the editor, facilitating rapid development and debugging.

Moreover, VS Code's debugging capabilities are invaluable for Python developers, with features like breakpoints, variable inspection, and step-through debugging making it easy to identify and fix issues in Python code. Integration with virtual environments and package management tools like pipenv and Poetry further streamlines Python development workflows.

- **PYTHON**

Python is a versatile and high-level programming language known for its simplicity, readability, and ease of use. It is widely used in various domains such as web development, data analysis, artificial intelligence, scientific computing, and automation.

Python's syntax is concise and intuitive, making it easy for beginners to learn and understand. Its indentation-based structure promotes clean and readable code, enhancing collaboration and maintainability. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming, providing flexibility for different coding styles and project requirements.

One of Python's key strengths is its extensive standard library, which includes modules for a wide range of tasks such as file I/O, networking, database access, and regular expressions. This rich ecosystem reduces the need for external dependencies and simplifies development workflows.

Python's popularity in data science and machine learning is attributed to libraries such as NumPy, pandas, matplotlib, and scikit-learn, which provide powerful tools for data manipulation, analysis, visualization, and machine learning model development. These libraries leverage Python's simplicity and expressiveness to enable efficient data-driven decision-making and predictive analytics

Python's cross-platform compatibility allows code written in Python to run seamlessly on various operating systems, including Windows, macOS, and Linux. Its vibrant community and extensive documentation provide ample resources and support for developers of all levels.

- **CX_FREEZE AND INNO SETUP**

Cx_Freeze and Inno Setup are two popular tools used in Python development for different purposes:

Cx_Freeze: It's a set of scripts and modules for freezing Python scripts into executables. This means you can take a Python script and package it into a standalone executable that can run on machines that don't have Python installed. Cx_Freeze works by analyzing your script's dependencies, including all imported modules, and bundling them together into a single executable file.

Inno Setup: It's a free installer for Windows programs. It creates installers for your software that can handle the installation process on end-user machines. Inno Setup scripts define how your software should be installed, including specifying files to be copied, creating shortcuts, adding registry entries, and more. It's particularly useful for creating professional-looking installation wizards for your applications.

So, to use them together, you would typically use Cx_Freeze to freeze your Python script into an executable and then use Inno Setup to create an installer package for distributing your application. This installer would include the frozen executable along with any other necessary files and instructions for installing the application on a user's system. This combination is often used in Windows development to create distributable packages for Python applications.

- **SOME IMPORTANT PYTHON PACKAGES**

1.TENSORFLOW:

TensorFlow is an open-source machine learning framework developed by Google. It is widely used for building and training deep learning models, including neural networks, for a variety of tasks such as image classification, natural language processing, and reinforcement learning.

TensorFlow provides a flexible and comprehensive ecosystem for developing machine learning applications, with features including:

TensorFlow Core: A low-level API for building and training machine learning models, with support for tensors, mathematical operations, and automatic differentiation.

Keras API: A high-level API that simplifies the process of building and training neural networks, making it accessible to beginners and experts alike.

TensorBoard: A visualization toolkit for visualizing and debugging TensorFlow models, including tools for monitoring training metrics, visualizing model architectures, and analyzing performance.

TensorFlow Hub: A library for reusable machine learning modules, including pre-trained models and embeddings, enabling developers to leverage existing models for their applications.

2. SCIKIT-LEARN:

scikit-learn is a popular machine learning library in Python that provides simple and efficient tools for data mining and analysis. It is built on top of NumPy, SciPy, and matplotlib, making it easy to integrate with other scientific computing libraries in Python.

scikit-learn offers a wide range of machine learning algorithms and utilities, including:

- Supervised Learning: Algorithms for classification, regression, and ensemble methods.
- Unsupervised Learning: Algorithms for clustering, dimensionality reduction, and outlier detection.
- Model Evaluation: Tools for evaluating model performance using metrics such as accuracy, precision, recall, and F1-score.
- Model Selection: Techniques for model selection and hyperparameter tuning, including cross-validation and grid search.

3. TKINTER

Tkinter is a standard GUI library for Python that provides a simple and intuitive way to create desktop applications with graphical interfaces. It comes pre-installed with Python, making it readily available for developers to use.

With tkinter, developers can create windows, buttons, labels, entry fields, and other GUI components to build interactive applications. Its event-driven programming model allows developers to define event handlers that respond to user actions such as button clicks or keyboard input.

Tkinter provides a wide range of widgets and layout managers for organizing and styling GUI elements, enabling developers to create visually appealing and functional applications. It also supports theming and customization options to tailor the look and feel of the application to suit specific requirements.

4.OPENCV:

OpenCV (Open Source Computer Vision Library) is a powerful open-source library for computer vision and image processing tasks. It provides a comprehensive set of tools and algorithms for analyzing and manipulating images and videos.

With OpenCV, developers can perform a wide range of computer vision tasks, including image filtering, edge detection, object detection, facial recognition, motion tracking, and more. It offers efficient implementations of popular algorithms and techniques, making it suitable for real-time applications.

OpenCV supports various image formats and provides functions for loading, saving, and displaying images and videos. It also integrates seamlessly with other Python libraries such as NumPy and matplotlib, facilitating data manipulation and visualization tasks.

2. LITRATURE SURVEY

2.1 HISTORY OF PARKINSON DISEASE

The history of Parkinson's Disease (PD) detection software parallels the broader advancements in technology and medical research aimed at improving the diagnosis and management of the disease. Here's an overview of the evolution of PD detection software:

- Early Clinical Tools: In the early stages, PD diagnosis relied primarily on clinical observation and assessment by healthcare professionals. While there weren't dedicated software programs for PD detection, clinicians used paper-based rating scales like the Unified Parkinson's Disease Rating Scale (UPDRS) to evaluate motor symptoms and track disease progression.
- Emergence of Computerized Assessment Tools: With the advent of computers and digital technology, researchers began developing computerized assessment tools to aid in PD detection and monitoring. These tools digitized traditional rating scales and introduced objective measures for quantifying motor symptoms, such as tremor amplitude, gait parameters, and bradykinesia.
- Wearable Sensors and Mobile Applications: The proliferation of wearable sensors and mobile health applications revolutionized PD detection and monitoring. Wearable devices, equipped with accelerometers and gyroscopes, can capture movement data in real-time, providing insights into motor fluctuations and dyskinesias. Mobile applications enable remote monitoring, symptom tracking, and communication between patients and healthcare providers.
- Machine Learning and Artificial Intelligence: Machine learning and artificial intelligence (AI) techniques have been increasingly applied to PD detection software. These algorithms analyze large datasets of movement data to identify patterns, classify motor symptoms, and predict disease progression. AI-powered software holds promise for early detection, personalized treatment optimization, and predictive analytics in PD management
- Virtual Reality and Gamification: Virtual reality (VR) and gamification techniques are being explored as novel approaches to PD detection and rehabilitation. VR-based assessments simulate real-life scenarios to evaluate balance, gait, and motor coordination, providing a more immersive and engaging experience for patients. Gamified exercises encourage adherence to rehabilitation programs and enable continuous monitoring of motor function.
- Remote Monitoring Platforms: Integrated remote monitoring platforms combine various technologies, including wearable sensors, mobile applications, and telemedicine, to enable comprehensive and continuous monitoring of PD patients in their home environments. These platforms facilitate early detection of motor fluctuations, medication adherence, and timely intervention by healthcare providers.

2.2 EXISTING MODEL OF PARKINSON DISEASE DETECTION

2.2.1 NQ MEDICAL -

nQ Medical is a company that has developed innovative software for the detection and monitoring of neurological conditions, including Parkinson's Disease (PD). The primary focus of nQ Medical's technology is on analyzing the fine motor control involved in typing on a keyboard. This involves capturing the timing and patterns of key presses and releases as users type.

The software extracts digital biomarkers from typing patterns. These biomarkers are indicators of neurological function and can reflect the subtle motor impairments characteristic of PD.

Integration and User Experience:

The software can be integrated into existing devices such as laptops, tablets, and smartphones without requiring specialized hardware. nQ Medical emphasizes data security and privacy. All data collected is anonymized and encrypted to protect user information. The software aims to detect PD at an early stage, potentially before clinical symptoms become apparent.

By leveraging everyday technology and sophisticated machine learning algorithms, it offers a non-invasive, cost-effective, and accessible tool for early diagnosis and continuous monitoring.

Advantages of nQ Medical's Software

- **Non-Invasive**: The software monitors keystroke dynamics, a non-invasive method that does not require any special equipment or procedures. Users continue their normal typing activities, making it easy to implement.
- **Seamless Integration**: It can be integrated into existing devices like laptops, tablets, and smartphones without requiring additional hardware. This makes the technology accessible and easy to use for a wide range of use

Objective Data:

- **Quantitative Analysis**: Provides objective, quantitative data on motor function, which can be more reliable than subjective assessments.
- **Longitudinal Tracking**: Enables longitudinal tracking of disease progression, allowing for more precise adjustments to treatment plans.

Drawbacks of nQ Medical's Software

- **Specific Focus**: The software focuses primarily on fine motor control related to typing. It may not capture other important symptoms of PD, such as gait abnormalities, speech changes, or non-motor symptoms.

- **Privacy Concerns**: The continuous collection of typing data raises privacy and security concerns. Ensuring that data is anonymized, encrypted, and securely stored is crucial to protect user information.
- **User Consent**: Obtaining informed consent from users for continuous monitoring and data collection can be challenging and may affect participation rates.

nQ Medical™
nQ Cortex
The nQ platform is an integrated solution for brain health screening and monitoring, with products designed for use in research, clinical care and daily life

Fig 1: NQ MEDICAL SOFTWARE

Fig 2: PREDICT-PD

2.2.2 PREDICT-PD

PREDICT-PD is an ongoing research project based in the UK that aims to identify early markers of Parkinson's Disease (PD) through a combination of clinical assessments, genetic testing, and the analysis of various biomarkers. The goal is to predict the risk of developing PD before the onset of motor symptoms, allowing for earlier intervention and improved outcomes.

Advantages of PREDICT-PD

- **Early Detection**: By identifying individuals at high risk for PD before the onset of motor symptoms, PREDICT-PD enables earlier intervention, potentially slowing disease progression and improving quality of life.
- **Comprehensive Approach**: The combination of genetic, clinical, cognitive, and olfactory assessments provides a holistic view of the factors contributing to PD risk.
- **Non-Invasive**: Most assessments and sample collections are non-invasive, making it easier for participants to engage in the study.

Drawbacks of PREDICT-PD

- **False Positives/Negatives**: Predictive models may produce false positives (identifying individuals at risk who do not develop PD) or false negatives (failing to identify individuals who do develop PD). This can lead to unnecessary anxiety or missed early intervention opportunities.
- **Generalizability**: The study population may not be representative of the general population, potentially limiting the generalizability of the findings.

- **Data Privacy and Security**: The collection and storage of sensitive genetic and health data raise concerns about data privacy and security. Ensuring robust data protection measures is crucial.

2.3 PROPOSED SYSTEM/SOLUTION

Parkinson's Disease (PD) is a chronic and progressive neurodegenerative disorder that significantly impacts the motor system and overall quality of life for affected individuals. The primary symptoms include tremors, rigidity, bradykinesia, and postural instability. Early detection and personalized management are crucial for improving patient outcomes and slowing disease progression. Traditional approaches often rely on clinical assessments and symptomatic treatments, which can be subjective and may not capture the nuanced progression of the disease.

This project proposes a comprehensive, software-based system that integrates advanced technologies to redefine PD management. The system aims to leverage handwriting analysis, voice input analysis, body movement analysis, and real-time fall detection to provide a multifaceted approach to PD care. The proposed solution is designed to be non-invasive, using readily available digital tools and machine learning algorithms to analyze data and provide actionable insights.

2.3.1. Objectives

- Develop a comprehensive framework for early detection and personalized management of PD.
- Utilize handwriting analysis, voice input analysis, and body movement analysis as primary modalities for PD detection and classification.
- Implement advanced machine learning algorithms, including Convolutional Neural Networks (CNN) for handwriting analysis and XGBoost for voice input analysis.
- Provide personalized recommendations tailored to each patient's unique presentation to optimize therapeutic outcomes and enhance patient well-being.
- Ensure seamless integration of real-time fall detection to enhance patient safety and caregiver response.

2.3.2. System Architecture

The proposed system comprises four primary components:

- Handwriting Analysis Module
- Voice Input Analysis Module
- Body Movement Analysis Module
- Real-time Fall Detection Module

Each module is designed to function independently while contributing to a unified framework that provides comprehensive PD management.

2.3.3. Handwriting Analysis Module

a. Objective: Detect early signs of PD and classify disease stages through handwriting analysis.

b. Methodology:

- **Data Collection:** Collect handwritten samples from individuals with PD and healthy controls using digital tablets or scanned documents. Digital tablets provide a consistent and detailed data collection method, capturing pressure, speed, and stroke patterns.
- **Feature Extraction:** Utilize Convolutional Neural Networks (CNN) to extract features such as writing pressure, speed, stroke length, and micrographia. The CNN model identifies subtle deviations in handwriting that may indicate the presence of PD.
- **Model Training:** Train the CNN model on a large dataset comprising thousands of handwritten samples. Use transfer learning techniques to fine-tune the model, improving its accuracy and generalizability across different handwriting styles.
- **Classification:** Develop algorithms to classify handwriting samples, enabling early detection and staging of PD. The classification process involves analyzing the extracted features and comparing them with known patterns associated with different stages of PD.

c. Output:

- Early detection of PD through handwriting abnormalities.
- Classification of disease stages based on handwriting characteristics.
- Detailed reports highlighting specific handwriting features that indicate PD progression.

2.3.4. Voice Input Analysis Module

a. Objective:

Identify speech abnormalities and tremors associated with PD through voice analysis.

b. Methodology:

- **Data Collection:** Record voice samples from individuals with PD and healthy controls during structured speech tasks, such as reading passages or spontaneous conversation. These tasks are designed to elicit speech patterns affected by PD.
- **Feature Extraction:** Extract acoustic features such as pitch, intensity, formant frequencies, jitter, and shimmer. These features provide insights into the vocal characteristics and speech abnormalities commonly seen in PD patients.
- **Model Training:** Employ the XGBoost algorithm to train on the extracted features. XGBoost is chosen for its robustness and ability to handle high-dimensional data effectively.

- **Real-time Analysis:** Implement real-time voice analysis for continuous monitoring. The system captures and analyzes voice input during regular interactions, providing immediate feedback on speech-related symptoms.

c. Output:

- Detection of speech abnormalities indicative of PD, such as monotone speech, reduced volume, and tremors in the voice.
- Classification of disease severity based on voice characteristics.
- Real-time monitoring and alerts for significant changes in speech patterns.

2.3.5. Body Movement Analysis Module

a. Objective: Detect motor symptoms such as bradykinesia through body movement analysis.

b. Methodology:

- **Data Collection:** Capture video recordings of movement tasks using standard cameras or webcam feeds. Tasks walking to highlight PD symptoms.
- **Feature Extraction:** Utilize the MediaPipe framework to extract skeletal pose features from video data. MediaPipe's pose estimation model accurately tracks body movements and joint positions.
- **Real-time Analysis:** Implement real-time movement analysis for continuous monitoring. The system processes video feeds in real time, providing immediate feedback on motor function.

c. Output:

- Detection of motor symptoms indicative of PD
- Classification of disease progression based on movement patterns.
- Real-time monitoring and alerts for significant changes in motor function.

2.3.6 Real-time Fall Detection Module

a. Objective: Provide real-time detection and alerts for falls to enhance patient safety.

b. Methodology:

- **Data Collection:** Use video streams or webcam feeds to monitor patient movement continuously. The system captures normal daily activities and detects anomalies indicative of a fall.
- **Feature Extraction:** Employ computer vision techniques to analyze posture and detect falls. The system tracks key body points and recognizes rapid changes in position that suggest a fall.
- **Model Training:** Train machine learning models, such as KNN to classify fall events based on visual cues which allows the system to accurately detect falls and reduce false positives.

- **Real-time Alerts:** Trigger immediate alerts to caregivers or emergency services through mobile or web-based applications in case of a fall. The alert system is designed to provide location and situation details, enabling quick response.

c. Output:

- Real-time detection of falls, enhancing patient safety.
- Immediate alerts to caregivers or emergency contacts, reducing response time.
- Continuous monitoring, providing peace of mind to patients and their families.

2.3.7. Integration and User Interface

a. User Interface:

- Design user-friendly interfaces for patients and caregivers. The interface includes intuitive controls, visualizations of health metrics, and easy access to alerts and recommendations.
- Provide features for real-time monitoring, alerts, and detailed reports. The system should be accessible via web and mobile applications.

2.3.8. Evaluation and Validation

- Measure performance using standardized metrics, such as accuracy, sensitivity, specificity, precision, recall, and F1-score, to ensure the system meets clinical standards.
- Continuously refine models based on feedback and performance data to improve accuracy and reliability.

2.3.9 Expected Impact

- **Improved Early Detection and Diagnosis:** The system's advanced analysis capabilities enable early detection of PD, leading to timely intervention and better patient outcomes.
- **Enhanced Safety and Quality of Life:** Real-time monitoring and fall detection improve patient safety and provide reassurance to patients and caregivers.
- **Contribution to PD Research:** The innovative methodologies and comprehensive data collection contribute to advancing PD research and clinical practice.
- **Scalability and Adaptability:** The framework can be adapted for other neurodegenerative disorders or healthcare applications, demonstrating its versatility and scalability.

The proposed system represents a significant advancement in PD management, leveraging advanced software-based technologies for early detection, accurate classification, and personalized care. By integrating handwriting, voice, and body movement analysis with real-time fall detection, the system provides a holistic approach to addressing the complexities of Parkinson's Disease. This innovative solution has the potential to improve patient outcomes, enhance quality of life, and contribute to the broader goal of advancing neurological healthcare. Through collaboration with healthcare professionals and continuous refinement, the project aims to make a meaningful impact in the field of neurology and improve the lives of individuals living with Parkinson's Disease.

2.4 FEASIBILITY STUDY

The feasibility study evaluates the practicality of developing and implementing the proposed software-based system for managing Parkinson's Disease (PD). This study assesses the technical, operational, economic, and legal aspects to ensure the project's success.

2.4.1. Technical Feasibility

a. Technology and Tools:

- **Handwriting Analysis:** Convolutional Neural Networks (CNN) are mature and widely used for image recognition tasks, making them suitable for handwriting analysis. Digital tablets and scanners are readily available for data collection.
- **Voice Input Analysis:** XGBoost is a robust and scalable algorithm for classification tasks. Acoustic feature extraction tools (e.g., Praat) are well-established and can be integrated into the system.
- **Body Movement Analysis:** MediaPipe provides advanced pose estimation capabilities, and machine learning models (e.g., SVM, Random Forests) are effective for classifying movement patterns.
- **Real-time Fall Detection:** Computer vision techniques using KNN are effective for detecting falls from video streams. Modern cameras and webcams provide adequate video quality for analysis.

b. Infrastructure:

- Hardware Specifications

- **Memory (RAM):** At least 8GB - Smooth operation, handles multimedia data efficiently.
- **Processor (CPU):** Intel Core i5+ / AMD Ryzen 5+ - Handles video/audio processing, ensures responsiveness.
- **Storage:** 256GB SSD - Fast data access, sufficient for OS and multimedia.
- **Camera:** Standard webcam - Suitable for video capture.
- **Microphone:** Standard microphone - Suitable for audio capture.
- **Operating System:** Windows 7+ (64 Bit), macOS, Linux - Flexible compatibility.

- Software Specifications

- **VS Code:** Syntax highlighting, code completion, IntelliSense, debugging.
- **Python:** Simple syntax, extensive libraries, cross-platform.
- **Cx_Freeze and Inno Setup:** Packages Python scripts into executables, creates Windows installers.

c. Expertise:

- **Development Team:** A multidisciplinary team with expertise in machine learning, computer vision, software engineering, and healthcare is essential. The availability of skilled professionals is feasible given the popularity of these fields.

2.4.2. Operational Feasibility

a. Implementation Plan:

- **Phase 1:** Research and data collection – Collaborate with healthcare institutions to gather initial datasets and refine system requirements.
- **Phase 2:** Development and testing – Build and test the individual modules (handwriting, voice, body movement, and fall detection).
- **Phase 3:** Integration and deployment – Integrate the modules into a unified system, conduct comprehensive testing, and deploy in clinical settings.
- **Phase 4:** Monitoring and refinement – Continuously monitor system performance, gather user feedback, and refine algorithms and features.

b. User Training and Support:

- Provide comprehensive training for healthcare professionals, patients, and caregivers on using the system.
- Establish a support team to address technical issues and provide ongoing assistance.

c. Adoption:

- Engage with stakeholders (healthcare providers, patients, caregivers) to promote adoption. Demonstrate the system's benefits through pilot programs and case studies.

d. Maintenance:

- Regular updates and maintenance are required to ensure the system remains effective and secure. Establish a dedicated team for ongoing support and improvements.

2.4.3. Legal Feasibility

a. Regulatory Compliance:

- Ensure the system complies with relevant healthcare regulations, such as HIPAA in the United States, to protect patient data privacy and security.
- Obtain necessary approvals from regulatory bodies for clinical use.

b. Data Security and Privacy:

- Implement robust data encryption, secure authentication, and access control measures to protect patient data.
- Establish clear data governance policies and procedures to ensure compliance with data protection regulations.

3. SYSTEM ANALYSIS & DESIGN

3.1 Specifications of Requirement

The Specifications of Requirement outline the detailed functional, non-functional, and system requirements for the proposed software-based system for managing Parkinson's Disease (PD). These specifications serve as a blueprint for the development, implementation, and maintenance of the system.

3.1.1. Functional Requirements

a. Handwriting Analysis Module

Data Input:

- Accepts handwritten samples from digital tablets or scanned documents.
- Supports various file formats (e.g., PNG, JPG, JPEG).

Classification:

- Classifies handwriting samples to detect PD and stage the disease.
- Provides detailed reports on handwriting characteristics.

User Interface:

- Allows users to upload handwritten samples.
- Displays classification results and detailed analysis.

b. Voice Input Analysis Module

Data Input:

- Records voice samples during structured speech tasks (e.g., reading passages, spontaneous conversation).
- Supports various audio formats (e.g., WAV, MP3).

Feature Extraction:

- Extracts acoustic features such as pitch, intensity, formant frequencies, jitter, and shimmer.

Classification:

- Uses XGBoost algorithm for voice input classification.
- Detects speech abnormalities indicative of PD and classifies disease severity.

User Interface:

- Allows users to record and upload voice samples.
- Displays analysis results and alerts for speech abnormalities.

c. Body Movement Analysis Module

Data Input:

- Captures video recordings of movement tasks using standard cameras or webcams.
- Supports various video formats (e.g., MP4, AVI).

Feature Extraction:

- Utilizes MediaPipe framework to extract skeletal pose features from video data.

Real-time Analysis:

- Provides real-time movement analysis for continuous monitoring.

User Interface:

- Allows users to upload video recordings.
- Displays analysis results and alerts for motor symptoms.

d. Real-time Fall Detection Module

Data Input:

- Monitors video streams or webcam feeds for continuous patient movement.
- Supports live video streaming.

Feature Extraction:

- Employs computer vision techniques to analyze posture and detect falls.

Classification:

- Uses KNN to classify fall events.

Real-time Alerts:

- Triggers immediate alerts to caregivers or emergency services in the event of a fall.
- Provides location and situation details in alerts.

User Interface:

- Allows users to enable and configure fall detection settings.
- Displays real-time monitoring status and alert history.

3.1.2. Non-Functional Requirements

a. Performance:

- The system must process data inputs and generate analysis results within a reasonable timeframe (e.g., within 5 seconds for real-time monitoring).
- The system should handle concurrent users without significant degradation in performance.

b. Scalability:

- The system must be scalable to accommodate increasing amounts of data and users.
- Cloud-based infrastructure should be used to support scalability.

c. Usability:

- The user interface should be intuitive and user-friendly.
- Provide comprehensive user documentation and training materials.

3.1.3. System Requirements

Hardware Specifications

- **Memory (RAM):** At least 8GB - Smooth operation, handles multimedia data efficiently.
- **Processor (CPU):** Intel Core i5+ / AMD Ryzen 5+ - Handles video/audio processing.
- **Storage:** 256GB SSD - Fast data access, sufficient for OS and multimedia.
- **Camera:** Standard webcam - Suitable for video capture.
- **Microphone:** Standard microphone - Suitable for audio capture.
- **Operating System:** Windows 7+ (64 Bit), macOS, Linux - Flexible compatibility.

Software Specifications

- **VS Code:** Syntax highlighting, code completion, IntelliSense, debugging.
- **Python:** Simple syntax, extensive libraries, cross-platform.
- **Cx_Freeze and Inno Setup:** Packages Python scripts into executables, creates Windows installers.

Important Python Packages

- **TensorFlow:** For building/training deep learning models.
- **scikit-learn:** For data mining and analysis.
- **tkinter:** For creating desktop GUIs.
- **OpenCV:** For computer vision and image processing.

3.2 FLOW CHART OF THE SOFTWARE

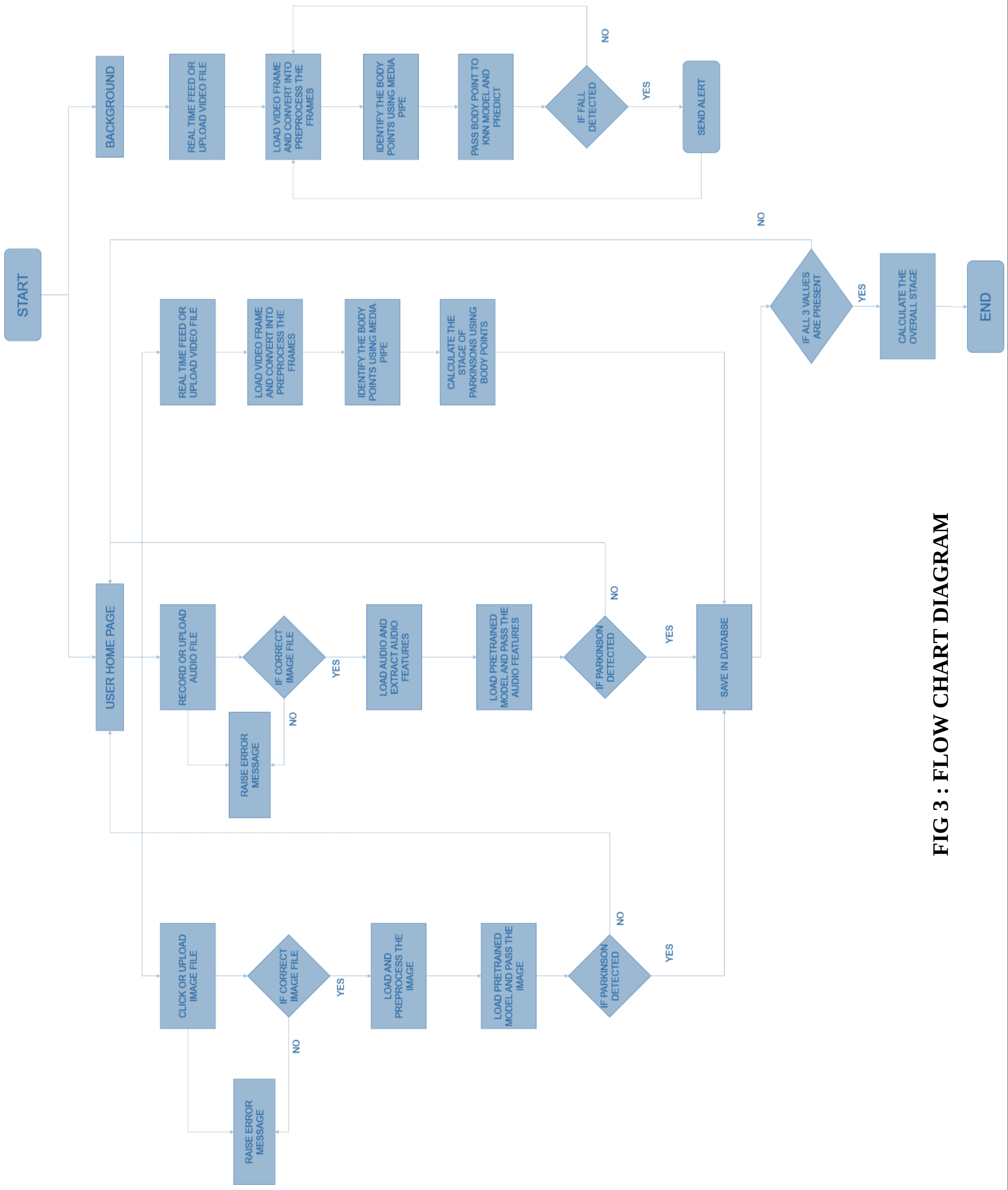

START
BACKGROUND
REAL TIME FEED OR UPLOAD VIDEO FILE
LOAD VIDEO FRAME AND CONVERT INTO PREPROCESS THE FRAMES
IDENTIFY THE BODY POINTS USING MEDIA PIPE
PASS BODY POINT TO KNN MODEL AND PREDICT
IF FALL DETECTED
NO
YES
SEND ALERT
USER HOME PAGE
REAL TIME FEED OR UPLOAD VIDEO FILE
LOAD VIDEO FRAME AND CONVERT INTO PREPROCESS THE FRAMES
IDENTIFY THE BODY POINTS USING MEDIA PIPE
CALCULATE THE STAGE OF PARKINSONS USING BODY POINTS
RECORD OR UPLOAD AUDIO FILE
IF CORRECT IMAGE FILE
YES
NO
RAISE ERROR MESSAGE
LOAD AUDIO AND EXTRACT AUDIO FEATURES
LOAD PRETRAINED MODEL AND PASS THE AUDIO FEATURES
IF PARKINSON DETECTED
NO
YES
SAVE IN DATABSE
CLICK OR UPLOAD IMAGE FILE
IF CORRECT IMAGE FILE
YES
NO
RAISE ERROR MESSAGE
LOAD AND PREPROCESS THE IMAGE
LOAD PRETRAINED MODEL AND PASS THE IMAGE
IF PARKINSON DETECTED
NO
YES
IF ALL 3 VALUES ARE PRESENT
NO
YES
CALCULATE THE OVERALL STAGE
END

FIG 3 : FLOW CHART DIAGRAM

3.3 DESIGN STEPS

3.3.1. Requirement Analysis:

Gather Requirements:

- Conduct interviews with healthcare professionals, patients, and caregivers to gather detailed requirements.
- Review existing literature and studies on Parkinson's Disease (PD) management to understand key features and functionalities needed.

Define Specifications:

- Document functional and non-functional requirements.
- Define system requirements including hardware, software, and network specifications.

3.3.2. System Architecture Design:

High-Level Architecture:

- Design the overall system architecture, including data flow and interaction between modules.
- Define the architecture layers (e.g., data layer, application layer, presentation layer).

Module Design:

- Design each module in detail (handwriting analysis, voice input analysis, body movement analysis, real-time fall detection).
- Define the inputs, processing logic, and outputs for each module.

3.3.3. Data Collection and Preprocessing:

Data Collection:

- Gather datasets for handwriting, voice, and body movement from clinical partners or public repositories.
- Ensure datasets are diverse and representative of the PD population.

Data Preprocessing:

- Clean and preprocess the data (e.g., normalize handwriting samples, filter noise from voice recordings, stabilize video feeds).
- Split data into training, validation, and test sets.

3.3.4. Model Development:

Model Selection:

- Choose appropriate machine learning models (e.g., CNN for handwriting, XGBoost for voice, MediaPipe with SVM/Random Forest for body movement).
- Extract relevant features from the collected data.
- Perform feature scaling and transformation as needed.

Model Training:

- Train models on the training data.
- Fine-tune hyperparameters using validation data.

3.3.5. Integration and Interface Design:

Integration:

- Integrate the individual modules into a cohesive system.
- Ensure smooth data flow and interoperability between modules.

User Interface Design:

- Design user-friendly interfaces for healthcare professionals, patients, and caregivers.
- Develop dashboards for data visualization and real-time monitoring.

3.3.6. System Implementation:

Development:

- Implement the designed system using chosen technologies and tools.
- Develop EXEs for seamless installation process

Testing:

- Conduct unit testing, integration testing, and system testing to ensure functionality.
- Perform user acceptance testing (UAT) with real users.

3.3.7. Deployment:

Deployment Planning:

- Plan the deployment strategy, including cloud infrastructure setup and data migration.

Deployment Execution:

- Deploy the system in a controlled environment initially for further testing.
- Roll out the system to production once it meets all quality standards.

3.3.8. Maintenance and Updates:

Ongoing Support:

- Establish a support team for system maintenance and user support.

Regular Updates:

- Continuously monitor system performance.
- Implement updates and improvements based on user feedback and new research.

3.3.9. Usability Testing:

Objective: Assess the system's ease of use and user satisfaction.

Criteria:

- Users should be able to complete tasks efficiently and without errors.
- Positive feedback from users regarding the system's usability.

3.3.10. Compatibility Testing:

Ensure the system works correctly across different devices and environments.

Test Steps:

- Test the system on various operating systems (Windows, macOS, Linux).
- Verify compatibility with different web browsers and mobile devices.

Criteria:

- The system should function consistently across all supported platforms and devices.
- No significant usability or functionality issues on any supported platform.

3.3.11. Continuous Monitoring and Feedback:

Establish processes for ongoing monitoring, feedback collection, and improvement.

- Implement monitoring tools to track system performance, usage metrics, and user feedback.
- Establish channels for users to report issues and provide suggestions for improvement.
- Regular monitoring should identify any performance degradation or issues promptly.
- User feedback should be addressed in a timely manner, and improvements should be implemented.

3.4 ALGORITHMS

3.4.1 Convolutional Neural Networks (CNNs):

This algorithm is used for handwriting analysis in the Parkinson disease detector. Convolutional Neural Networks (CNNs) have shown promising results in various image recognition tasks, including handwriting analysis. Here's a simplified explanation of how CNNs work for handwriting analysis:

- Data Preprocessing: Handwritten samples are digitized into images, where each pixel represents a point on the writing surface. These images are often grayscale or binary representations of the handwriting.
- Convolutional Layers: The CNN architecture consists of multiple layers, including convolutional layers. In these layers, small filters (kernels) are applied to the input image to extract features. These filters move across the image, detecting patterns such as edges, corners, and shapes.
- Pooling Layers: After each convolutional layer, pooling layers are typically applied to down sample the feature maps and reduce computational complexity. Common pooling operations include max pooling and average pooling, which retain the most prominent features from the previous layer.
- Fully Connected Layers: The output from the convolutional and pooling layers is flattened and fed into one or more fully connected layers. These layers learn complex patterns and relationships between the extracted features, enabling the network to make predictions.
- Activation Functions: Non-linear activation functions, such as ReLU (Rectified Linear Unit), are applied after each layer to introduce non-linearity into the network and enable it to learn complex relationships in the data.
- Training: The CNN is trained using a labeled dataset of handwritten samples. During training, the network adjusts its parameters (weights and biases) using optimization algorithms like gradient descent to minimize the difference between the predicted and actual labels.
- Loss Function: A loss function, such as categorical cross-entropy or mean squared error, is used to quantify the difference between the predicted and actual labels. The goal of training is to minimize this loss function.
- Validation and Testing: The trained CNN is evaluated on a separate validation set to assess its performance and fine-tune hyperparameters.

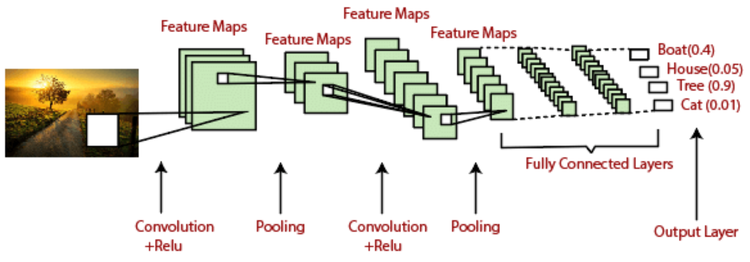


Fig 4: CNN ALGORITHM

3.4.2 MEDIA PIPE ALGORITHM

This algorithm is used for identifying the body movement. Mediapipe is an open-source framework developed by Google that offers solutions for various perception tasks, including body movement analysis. One of its key components is the Pose Detection Pipeline, which is designed to detect and track human body poses in real-time using machine learning models. Here's how the Mediapipe algorithm works for body movement analysis:

- Input: The input to the Mediapipe Pose Detection Pipeline is typically a video stream captured from a camera. This could be from a webcam, a smartphone camera, or any other device capable of capturing video input.
- Preprocessing: Before analyzing body movements, the input video frames may undergo preprocessing steps such as resizing, normalization, and color conversion to prepare them for analysis.
- Pose Detection Model: The heart of the Mediapipe algorithm is the Pose Detection Model, which is a convolutional neural network (CNN) trained to detect key points (or landmarks) on the human body, such as joints and body parts. The model is trained on a large dataset of annotated human poses to learn to accurately predict the locations of these key points in the input images.
- Key Point Detection: The Pose Detection Model analyzes each frame of the input video and predicts the locations of key points on the human body. These key points typically include joints such as the shoulders, elbows, wrists, hips, knees, and ankles, as well as other body parts like the nose and ears.
- Pose Estimation: Once the key points are detected, the algorithm constructs a skeletal representation of the human body by connecting the detected key points with lines to form body poses. These poses represent the spatial configuration of the human body in the input video frame.
- Tracking and Temporal Smoothing: To improve robustness and accuracy, the algorithm may incorporate techniques such as pose tracking and temporal smoothing. Pose tracking helps maintain consistency in pose estimation across consecutive frames, while temporal smoothing reduces jitter and noise in the estimated poses over time.
- Output: The output of the Mediapipe Pose Detection Pipeline is a sequence of detected body poses, typically represented as a series of key points and skeletal connections.

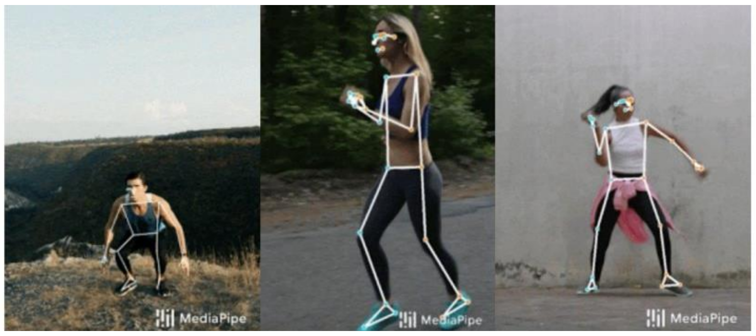


Fig 5: MEDIA PIPE

3.4.3 XGBOOST ALGORITHM

This algorithm is used for audio processing in the software. XGBoost (Extreme Gradient Boosting) is a powerful and efficient implementation of the gradient boosting framework for machine learning. It has been widely used for various tasks, including classification, regression, and ranking, due to its high performance and scalability. Here's an overview of XGBoost and how it can be applied to audio detection tasks:GBoost for Audio Detection Using XGBoost for audio detection involves several steps, from preprocessing the audio data to training the model and making predictions. Here's how it can be applied:

Audio Preprocessing:

- Feature Extraction: Convert raw audio signals into features that can be used by the XGBoost model. Common features include Mel-frequency cepstral coefficients (MFCCs), chroma features, spectral contrast, and zero-crossing rate. These features capture various aspects of the audio signal, such as frequency content and temporal changes.
- Normalization: Normalize the extracted features to ensure consistent scaling and to improve the performance of the model.
- Labeling: If the task is supervised learning (e.g., speech recognition, speaker identification, emotion detection), ensure the audio data is labeled appropriately.
- Train-Test Split: Split the dataset into training and testing sets to evaluate the model.

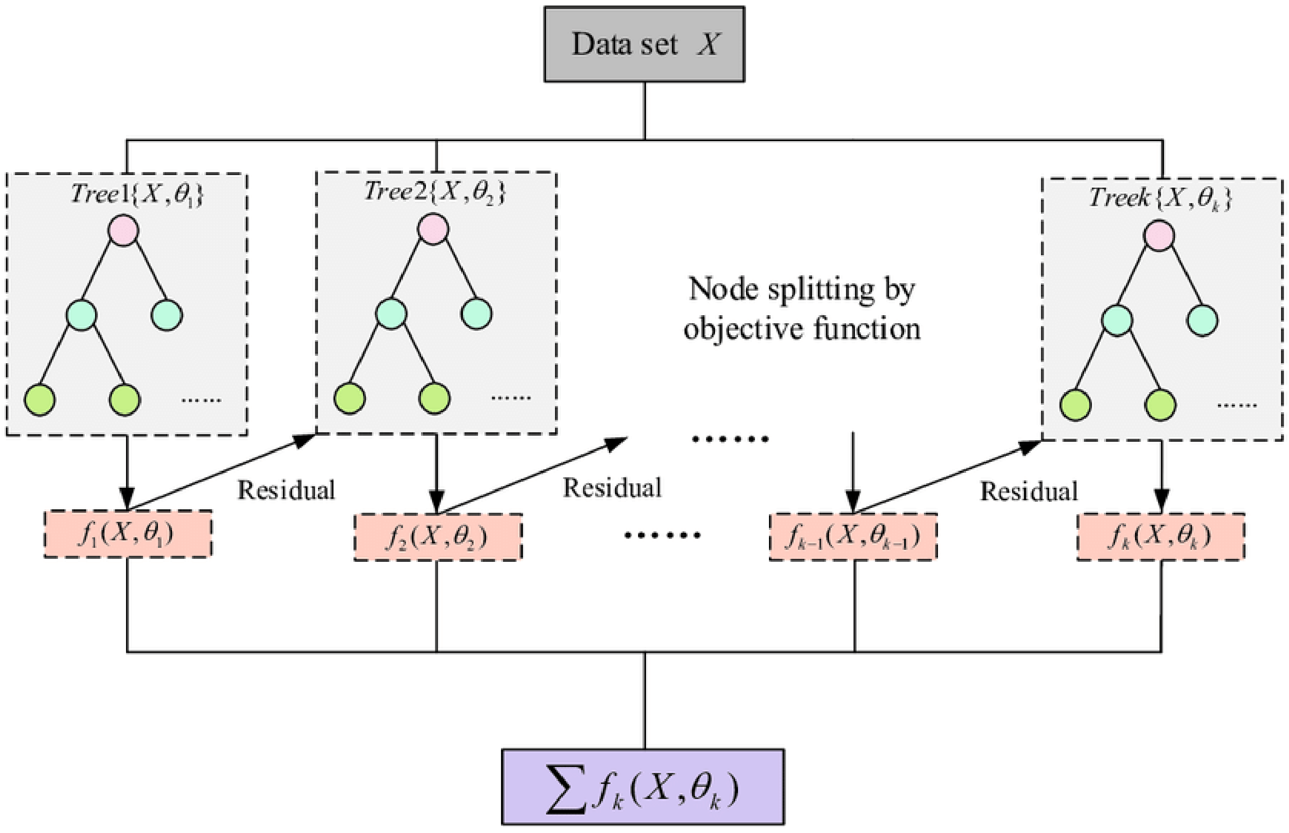


Fig 6: XG BOOST Algorithm

3.4.4 KNN ALGORITHM:-

The K-Nearest Neighbors (KNN) algorithm is a popular machine learning technique used for classification and regression tasks. It relies on the idea that similar data points tend to have similar labels or values. During the training phase, the KNN algorithm stores the entire training dataset as a reference. K-Nearest Neighbors (KNN) is a simple yet powerful machine learning algorithm often used for classification tasks, including fall detection.

Here’s how KNN works for fall detection:

- Data Sampling: Collect data at regular intervals to capture the dynamic nature of body movements. Each sample typically contains multiple features representing the sensor readings.
- Feature Extraction:
- Extract features from the raw sensor data that are indicative of falls. Common features include acceleration in different axes (x, y, z), magnitude of acceleration, angular velocity, and other derived metrics like tilt angles or velocity changes.
- Windowing: Use a sliding window approach to segment the continuous sensor data into smaller, manageable chunks. Each window can be represented as a feature vector.
- Fall Events: Label the segments of the data where a fall occurred as positive examples.
- Non-Fall Events: Label other segments as negative examples (e.g., normal activities like walking, sitting, standing).
- Model Construction:

  Define k: Choose the value of k, which represents the number of nearest neighbors to consider when making a prediction. The choice of k can significantly affect the performance of the algorithm.
- Distance Metric: Select an appropriate distance metric, such as Euclidean distance, to measure similarity between instances.

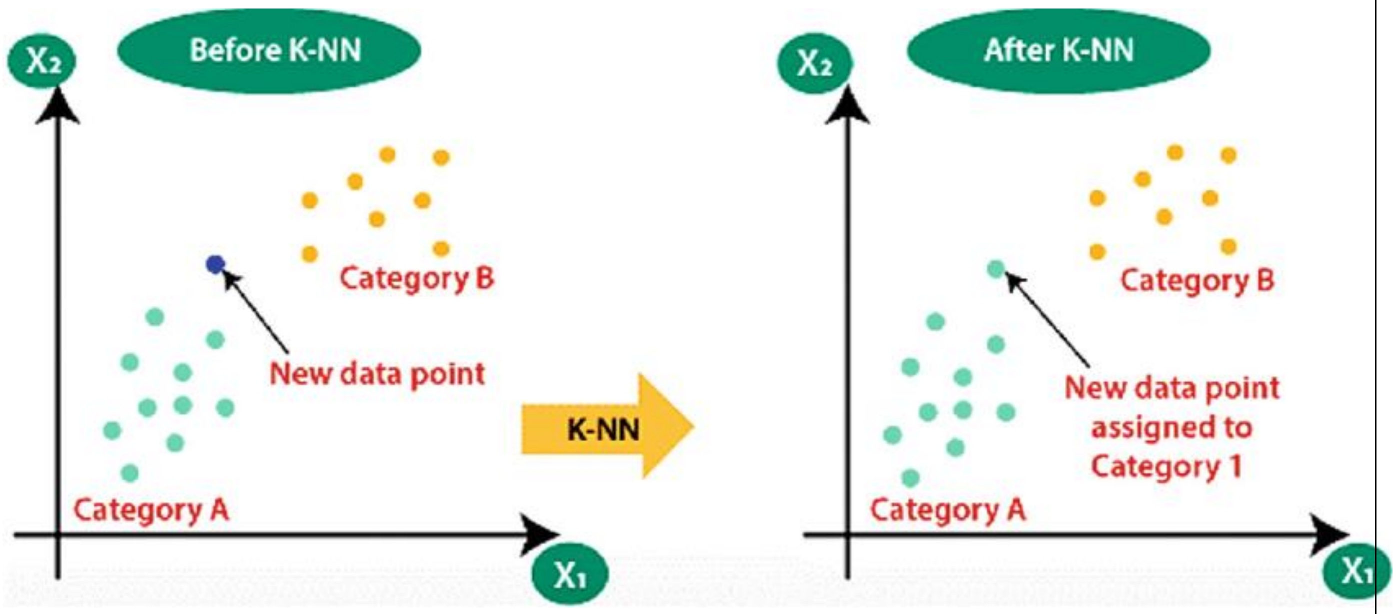


Fig 7: KNN Algorithm

3.5 PSEUDO CODE

3.5.1 Pseudo code of Main.py

```
Import required libraries and modules (tkinter, numpy, tensorflow, opencv, mediapipe, etc.)


Load pre-trained models:
    model1 = load_model('./Model/wave.h5')  # Model for wave handwriting task
    model2 = load_model('./Model/Spiral.h5')  # Model for spiral handwriting task
    new_model = load_xgboost_model('./Model/audio_model.json')    # XGBoost model for audio classification
Initialize dataoverall dictionary to store predictions from different modalities:
    dataoverall = {'spiral': None, 'wave': None, 'audio': None, 'video': None}


Define findDistance(x1, y1, x2, y2):
    # Calculate Euclidean distance between two points (x1, y1) and (x2, y2)
    dist = sqrt((x2 - x1)^2 + (y2 - y1)^2)
    Return dist


Define findAngle(x1, y1, x2, y2):
    # Calculate angle between two lines formed by (x1, y1), (x2, y2) and the y-axis
    theta = acos((y2 - y1) * (-y1) / (sqrt((x2 - x1)^2 + (y2 - y1)^2) * y1))
    degree = (180 / pi) * theta
    Return degree


Define mainvid(file_name):
    Initialize video capture from file_name or webcam
    Initialize variables for tracking frames and time
    For each frame:
        Read frame from video capture
        If frame is None, break the loop
        Convert frame to RGB
        Detect body keypoints using MediaPipe Pose
        Extract keypoint coordinates (left shoulder, right shoulder, left ear, left hip)
        Calculate offset, neck_inclination, and torso_inclination using findDistance and findAngle
        Classify Parkinson's stage based on neck_inclination and torso_inclination
        Draw keypoints and lines on the frame
        Display the frame
        If 'q' is pressed, break the loop
    Release video capture
    Determine overall prediction based on most frequent stage
    Return prediction and most frequent stage

Define predict_image_class1(img_array):
    # Predict for wave handwriting task
    Load image from img_array
    Convert to grayscale and resize to (120, 120)
    Normalize pixel values between 0 and 1
    Use model1 to predict class probabilities
    Get predicted class (Healthy or Parkinson) and confidence score
    Return prediction and confidence score

Define predict_image_class2(img_array):
    # Predict for spiral handwriting task
    Load image from img_array
    Convert to grayscale and resize to (120, 120)
    Normalize pixel values between 0 and 1
    Use model2 to predict class probabilities
    Get predicted class (Healthy or Parkinson) and confidence score
    Return prediction and confidence score

Define imgwin():
    Create GUI window for handwriting detection
    Define functions for button clicks:
        browse(): Open file dialog to select image
        predict_spiral(): Call predict_image_class2 on selected image
        predict_wave(): Call predict_image_class1 on selected image
        navigate(): Navigate to different window
        about(): Show information about the application
    Display GUI with buttons, image panel, and prediction results
    When image is uploaded, call predict_image_class1 and predict_image_class2
    Store predictions and confidence scores in dataoverall

Define audwin():
    Create GUI window for voice-based detection
    Define functions for button clicks:
        browse(): Open file dialog to select audio file
        predict_audio(): Extract audio features and use new_model to predict
        record_audio(): Record audio from microphone and save as temp.wav, then predict
        navigate(): Navigate to different window
        about(): Show information about the application
    Display GUI with buttons, image panel, and prediction results
    If audio file is uploaded, extract audio features and use new_model to predict
    If recording is requested, record audio and save as temp.wav, then predict
    Store prediction and confidence score in dataoverall

Define vidwin():
    Create GUI window for posture-based detection
    Define functions for button clicks:
        browse(): Open file dialog to select video file
        predict_video(): Call mainvid function on selected video file
        predict_webcam(): Call mainvid without arguments to use webcam
        navigate(): Navigate to different window
        about(): Show information about the application
    Display GUI with buttons, image panel, and prediction results
    If video file is uploaded, call mainvid function on the file
    If webcam prediction is requested, call mainvid without arguments
    Store prediction and stage in dataoverall

Define overall():
    Check if all modalities (spiral, wave, audio, video) have values in dataoverall
    If any modality is missing, show error message
    Initialize variables: parkval = 0, acc = 0, stage = 0
    For each modality:
        If prediction is 'Parkinson':
            Increment parkval
            Add confidence score to acc
        Else if modality is 'video':
            If prediction is 'Parkinson', increment parkval and store stage
    If parkval >= 2:
        Calculate average confidence score (av)
        Determine overall stage based on av and stage from video
        Display warning with overall stage
    Else:
        Display message indicating healthy person

Define homewin():
    Create main GUI window
    Define functions for button clicks to navigate to different modality windows
    Display GUI with buttons for different modalities and overall prediction
    When a modality button is clicked, open the corresponding window (imgwin, audwin, vidwin)
    When overall prediction button is clicked, call overall()

Define about():
    Show information about the application in a message box
Call homewin() to start the application.
```

3.5.2 PSEUDO CODE OF AUDIO.PY

```
function extract_audio_features(audio_file):
    # Load the audio file using librosa
    audio_signal, sampling_rate = load_audio_file(audio_file)

    features = []  # Initialize an empty list to store features

    # MDVP:Fo(Hz) - Fundamental frequency
    fundamental_frequencies = []
    for frame in split_audio_signal_into_frames(audio_signal):
        f0 = compute_fundamental_frequency(frame, min_freq=50, max_freq=400)
        fundamental_frequencies.append(f0)
    mean_fundamental_frequency = calculate_mean(fundamental_frequencies)
    features.append(mean_fundamental_frequency)

    # MDVP:Flo(Hz) - Lowest frequency of the acoustic signal
    lowest_frequency = calculate_min(fundamental_frequencies)
    features.append(lowest_frequency)

    # MDVP:Jitter(Abs) - Absolute jitter
    jitter_sequences = split_audio_signal_into_overlapping_frames(audio_signal)
    absolute_jitter = []
    for jitter_sequence in jitter_sequences:
        jitter_value = calculate_jitter(jitter_sequence)
        absolute_jitter.append(jitter_value)
    mean_absolute_jitter = calculate_mean(absolute_jitter)
    features.append(mean_absolute_jitter)

    # MDVP:Shimmer - Shimmer
    shimmer_sequences = split_audio_signal_into_overlapping_frames(audio_signal)
    shimmer_values = []
    for shimmer_sequence in shimmer_sequences:
        shimmer_value = calculate_shimmer(shimmer_sequence)
        shimmer_values.append(shimmer_value)
    mean_shimmer = calculate_mean(shimmer_values)
    features.append(mean_shimmer)

    # MDVP:Shimmer(dB) - Shimmer in dB
    root_mean_square_values = calculate_rms(audio_signal)
    mean_shimmer_db = calculate_mean(root_mean_square_values)
    features.append(mean_shimmer_db)

    # Shimmer:APQ3 - Amplitude Perturbation Quotient
    apq3 = calculate_amplitude_perturbation_quotient(3, root_mean_square_values)
    features.append(apq3)
```

```
# Shimmer:APQ5 - Amplitude Perturbation Quotient
spectral_centroids = calculate_spectral_centroids(audio_signal, sampling_rate)
apq5 = calculate_amplitude_perturbation_quotient(5, spectral_centroids)
features.append(apq5)

# MDVP:APQ - Amplitude Perturbation Quotient
spectral_bandwidths = calculate_spectral_bandwidths(audio_signal, sampling_rate)
apq = calculate_amplitude_perturbation_quotient(spectral_bandwidths)
features.append(apq)

# Shimmer:DDA - Shimmer (in DDA units)
spectral_flatness = calculate_spectral_flatness(audio_signal)
dda = calculate_mean(spectral_flatness)
features.append(dda)

# HNR - Harmonic-to-Noise Ratio
zero_crossing_rates = calculate_zero_crossing_rates(audio_signal)
hnr = calculate_harmonic_to_noise_ratio(zero_crossing_rates)
features.append(hnr)

# RPDE - Recurrence Period Density Entropy
tempograms = calculate_tempograms(audio_signal, sampling_rate)
rpde = calculate_recurrence_period_density_entropy(tempograms)
features.append(rpde)

# spread1
spectral_contrasts = calculate_spectral_contrasts(audio_signal, sampling_rate)
spread1 = calculate_mean(spectral_contrasts)
features.append(spread1)
# spread2
spectral_flatness2 = calculate_spectral_flatness(audio_signal)
spread2 = calculate_mean(spectral_flatness2)
features.append(spread2)
```

```
    # D2
    mel_spectrograms = calculate_mel_spectrograms(audio_signal, sampling_rate)
    d2 = calculate_mean(mel_spectrograms)
    features.append(d2)

    # PPE
    chroma_stfts = calculate_chroma_stfts(audio_signal, sampling_rate)
    ppe = calculate_mean(chroma_stfts)
    features.append(ppe)
    return features.
```

3.5.3 PSEUDO CODE OF FALL DETECTION .PY-

```
START PROGRAM

IMPORT required libraries (tkinter, ctypes, os, PIL, tkinter.messagebox)

CREATE a Tkinter window object 'home'
OPEN an image file 'images/home.png'
CREATE a PhotoImage object 'img' from the opened image
CREATE a Label widget 'panel' with the PhotoImage 'img'
PACK the 'panel' Label widget

GET the system screen dimensions (width and height) using user32.GetSystemMetrics()
CALCULATE the center position of the screen
SET the window title to "Fall Detection System"
SET the window geometry to position it at the center and set size to 900x566
DISABLE window resizing

DEFINE a function 'Exit'
    ASK for confirmation to exit using tkMessageBox.askquestion()
    IF user confirms ('yes')
        DESTROY the 'home' window
        EXIT the program
```

```
  ELSE
    SHOW an information message using tkMessageBox.showinfo()

DEFINE a function 'add'
  CREATE a new Tkinter window object 'root'
  SET the window title and geometry
  DISABLE window resizing
  SET the window background color
  CREATE a StringVar object 'eml'

  DEFINE a nested function 'ad'
    CHECK if the entered email contains '@'
    IF it does
      OPEN a file 'alert.txt' in write mode
      WRITE the entered email to the file
      CLOSE the file
      DESTROY the 'root' window
    ELSE
      SHOW a warning message using tkMessageBox.showinfo()

  CREATE a Label widget with the text "Add Email"
  CREATE an Entry widget 'a' with the StringVar 'eml'
  OPEN the 'alert.txt' file in read mode
  READ the file content and INSERT it into the Entry widget 'a'
  CREATE a Button widget with the text "ADD" and command 'ad'
  START the main event loop for 'root'

DEFINE a function 'st'
  CREATE a new Tkinter window object 'root'
  SET the window title and geometry
  DISABLE window resizing
  SET the window background color
  CREATE a StringVar object 'eml'
```

```
    DEFINE a nested function 'ad'
        GET the entered video directory path from the Entry widget 'a'
        IF the entered path is '0'
            SET the directory path to 0 (for realtime)
        IMPORT the 'Mediapipe_Pose' module
        CALL the 'start' function from 'Mediapipe_Pose' with the directory path

    CREATE a Label widget with the text "Video Directory (0 for realtime) :"
    CREATE an Entry widget 'a' with the StringVar 'eml'
    INSERT the default value '0' into the Entry widget 'a'
    CREATE a Button widget with the text "Start" and command 'ad'
    START the main event loop for 'root'

DEFINE a function 'contact'
    CREATE a new Toplevel window object 'contactwin'
    OPEN an image file 'images/about.png'
    CREATE a PhotoImage object 'img' from the opened image
    CREATE a Label widget 'panel' with the PhotoImage 'img'
    PACK the 'panel' Label widget

    GET the system screen dimensions (width and height) using user32.GetSystemMetrics()
    CALCULATE the center position of the screen
    SET the window title to "Fall Detection System"
    SET the window geometry to position it at the center and set size to 900x566
    DISABLE window resizing
    START the main event loop for 'contactwin'

OPEN an image file 'images/1.png'
CREATE a PhotoImage object 'img1' from the opened image
CREATE a Button widget 'b1' with the PhotoImage 'img1' and command 'st'
PLACE the Button widget 'b1' at (0, 149) position

OPEN an image file 'images/2.png'
CREATE a PhotoImage object 'img2' from the opened image
CREATE a Button widget 'b2' with the PhotoImage 'img2' and command 'add'
PLACE the Button widget 'b2' at (0, 228) position

OPEN an image file 'images/3.png'
CREATE a PhotoImage object 'img3' from the opened image
CREATE a Button widget 'b3' with the PhotoImage 'img3' and command 'contact'
PLACE the Button widget 'b3' at (0, 307) position

OPEN an image file 'images/4.png'
CREATE a PhotoImage object 'img4' from the opened image
CREATE a Button widget 'b4' with the PhotoImage 'img4' and command 'Exit'
PLACE the Button widget 'b4' at (0, 386) position

START the main event loop for 'home'

END PROGRAM
```

3.5.4 PSUDO CODE OF MEDIAPIPE POSE.PY-

```
mediapipe pose detect

Start Function(video_path):
    Import required libraries: cv2, mediapipe, numpy, time, joblib, datetime, smtplib, ssl
    Load the pre-trained pose classification model (pose_knn) using joblib
    Initialize drawing utilities for MediaPipe Pose (mp_drawing, mp_drawing_styles)
    Initialize MediaPipe Pose detection object (mp_pose)
    Initialize variables for tracking previous time (prevTime) and result points (res_point)
    Define a list of keypoint names (keyXYZ)
    Open the video capture object (cap) with the provided video_path
    Create a context for MediaPipe Pose detection (pose)
    Loop while the video capture is open:
        Read the next frame (image) from the video
        If the frame is not read successfully:
            Continue to the next iteration
        Make the image writable
        Convert the image from BGR to RGB color space
        Process the image with MediaPipe Pose detection (results)

        If pose landmarks are detected (results.pose_landmarks):
            For each landmark index and landmark object:
                Append the x, y, z coordinates of the landmark to res_point list

            Calculate the number of samples (shape1) based on the length of res_point and keyXYZ
            Reshape the res_point list into a 2D array with the shape (shape1, len(keyXYZ))
            Predict the class (fall or normal) using the loaded pose classification model (pose_knn)
            Clear the res_point list

            If the prediction is 'fall' (0):
                Display the text "Fall" on the image
                Construct the email message with the current date and time
                Send the email alert using SMTP server credentials and receiver email address

            Else:
                Display the text "Normal" on the image

        Make the image writable

        Convert the image back from RGB to BGR color space
        Draw the pose landmarks and connections on the image using mp_drawing utilities
        Calculate the current frame rate (fps) based on the time difference
        Display the current FPS on the image
        Show the processed image in a window
        If the 'Esc' key is pressed, break the loop
    Release the video capture object (cap.release())

    Close all the windows (cv2.destroyAllWindows()).
```

3.6 TESTING PROCESS

The testing process for a Parkinson's Disease (PD) detection system involves several stages to ensure the system is accurate, reliable, and ready for deployment. Below is a comprehensive outline of the testing process:

3.6.1 Unit Testing:

Objective: Verify the functionality of individual components or units of code.

Test Steps:

- Develop unit test cases for each function or method.
- Execute unit tests using a testing framework (e.g., Pytest, JUnit).
- Verify that each unit behaves as expected and produces the correct output.
- Mock external dependencies or use stubs as needed to isolate units for testing.

Criteria:

- Each unit test should pass without errors or failures.
- Test coverage should be comprehensive, covering all critical paths and edge cases.

3.6.2 Integration Testing:

Objective:

- Validate the interaction and communication between different modules or components.

Test Steps:

- Identify integration points between modules.
- Develop integration test cases to verify data flow and interactions.
- Execute integration tests to ensure seamless integration between modules.
- Verify that data is passed correctly between modules and that communication channels are functioning properly.

Criteria:

- Data should flow smoothly between integrated modules.
- Integration tests should pass without errors or failures.

3.6.2 System Testing:

Objective: Validate the complete system's functionality and behavior.

Test Steps:

- Develop system test cases covering all functional requirements.
- Execute system tests to verify that the system performs as expected.
- Test different use cases and scenarios to ensure comprehensive coverage.
- Verify system responses to user inputs and interactions.

Criteria:

- The system should meet all specified functional requirements.
- All system tests should pass without critical failures.

3.6.4 User Acceptance Testing (UAT):

Objective:

- Validate that the system meets end-users' expectations and requirements.

Test Steps:

- Define UAT scenarios based on user stories and use cases.
- Conduct UAT sessions with representative users, including healthcare professionals, patients, and caregivers.
- Gather feedback and observations from users during UAT sessions.
- Document and address any issues or usability concerns raised by users.

Criteria:

- Users should be able to perform tasks efficiently and without errors.
- Positive feedback from users regarding the system's usability and functionality.

3.6.5 Performance Testing:

Objective:

- Evaluate the system's performance under various load and stress conditions.

Test Steps:

- Define performance test scenarios to simulate expected user loads and peak traffic conditions.
- Measure system response times, throughput, and resource utilization under different load levels.
- Conduct stress tests to determine the system's breaking point and assess its resilience.

Criteria:

- The system should maintain acceptable performance metrics under expected user loads.
- It should gracefully degrade performance under stress conditions without crashing or freezing.

3.6.6 Security Testing:

Objective:

- Identify and address potential security vulnerabilities in the system.

Test Steps:

- Conduct security vulnerability assessments and penetration tests.
- Verify compliance with security standards and regulations (e.g., HIPAA).
- Test authentication and access control mechanisms.
- Review encryption protocols and data handling practices.

Criteria:

- No critical security vulnerabilities should be identified.
- The system should comply with all relevant security standards and regulations.

3.6.7 Regression Testing:

Objective:

- Ensure that new updates or changes do not introduce regressions or negatively impact existing functionality.

Test Steps:

- Re-run previously passed test cases after updates or changes.
- Verify that existing functionalities still perform correctly.
- Test both affected and unaffected areas of the system to detect any unintended side effects.

Criteria:

- All previously passed test cases should still pass.
- No new issues or regressions should be introduced by updates or changes.

By following these testing processes and criteria, the development team can ensure that the system meets all requirements, functions correctly, and provides value to users and stakeholders. Regular testing and validation are essential for maintaining system quality and reliability throughout its lifecycle.

4. RESULT AND OUTPUT

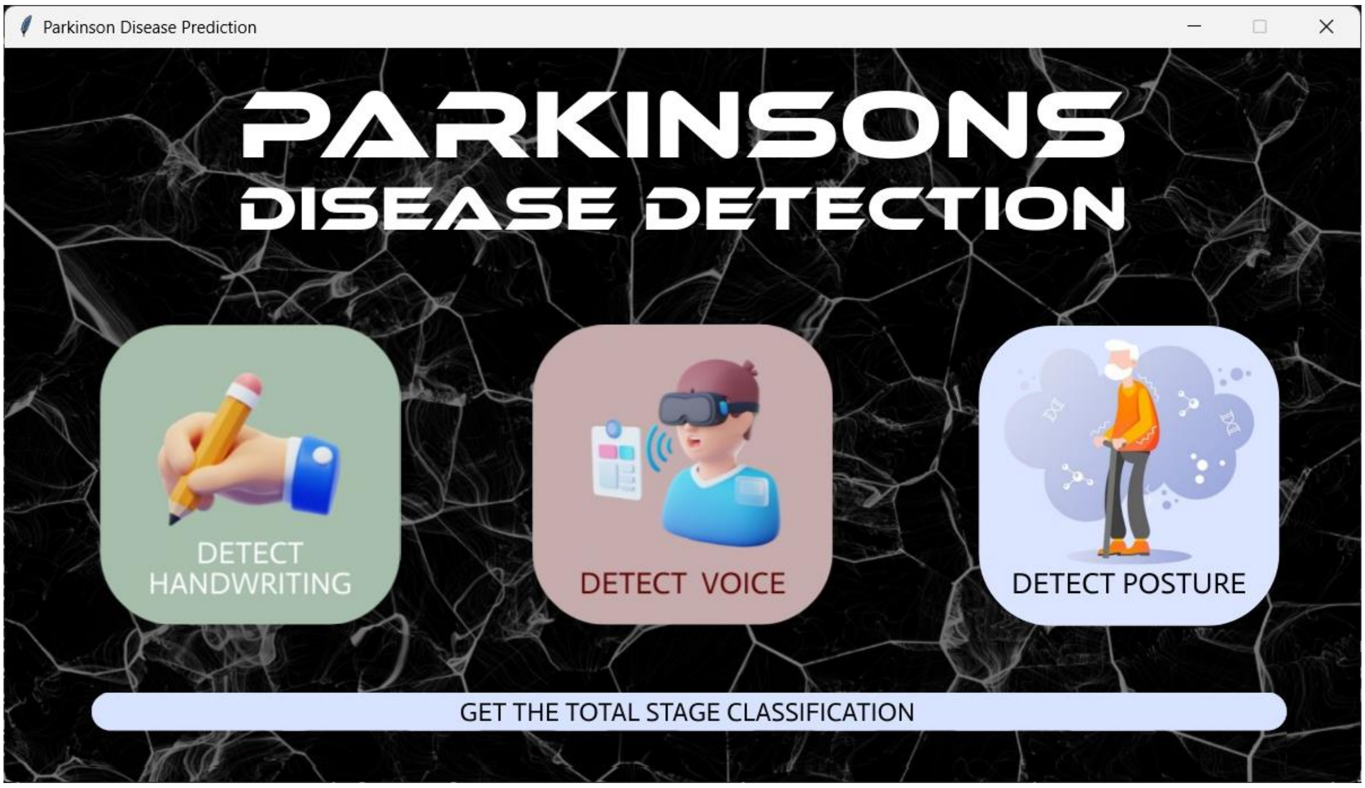


Fig 8: HOME PAGE OF SOFTWARE

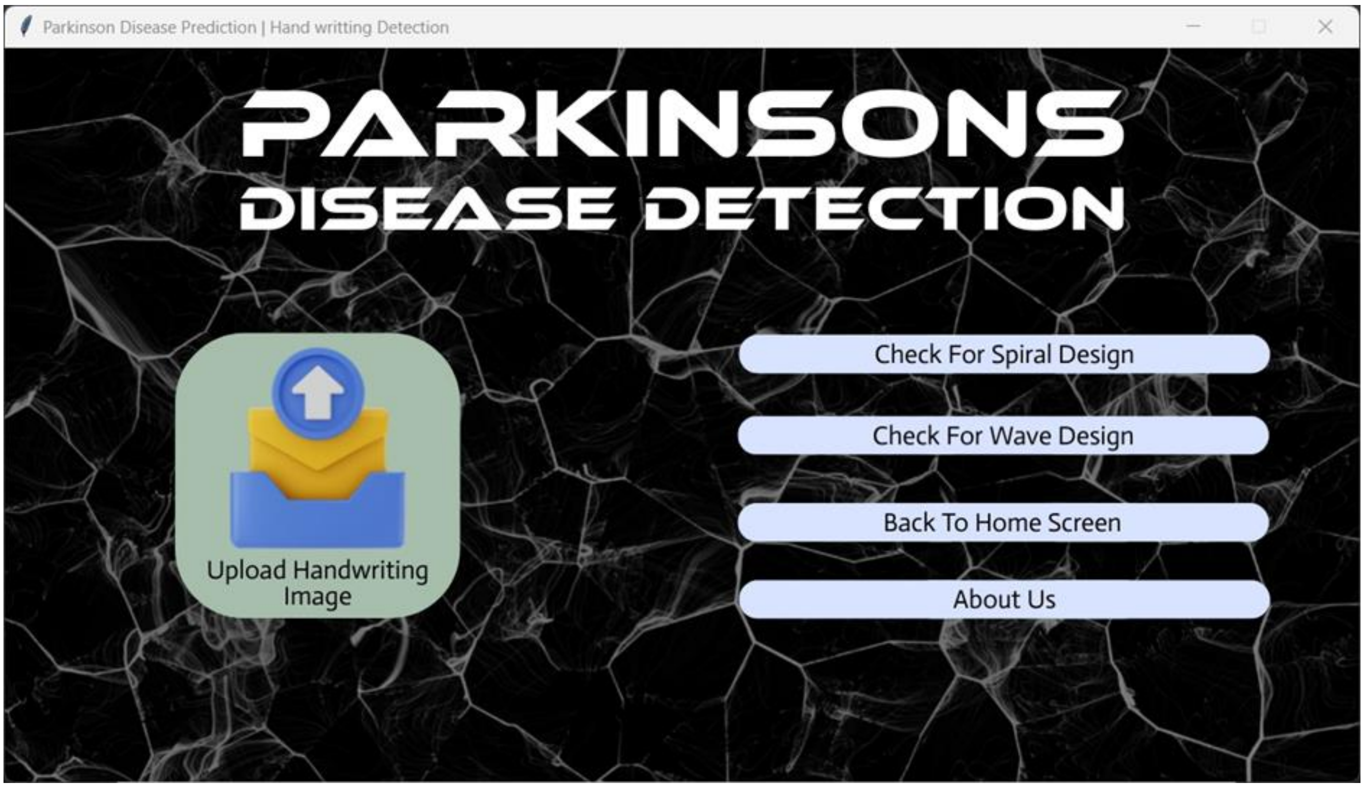


FIG 9: HAND WRITING BASED PREDICTION PAGE

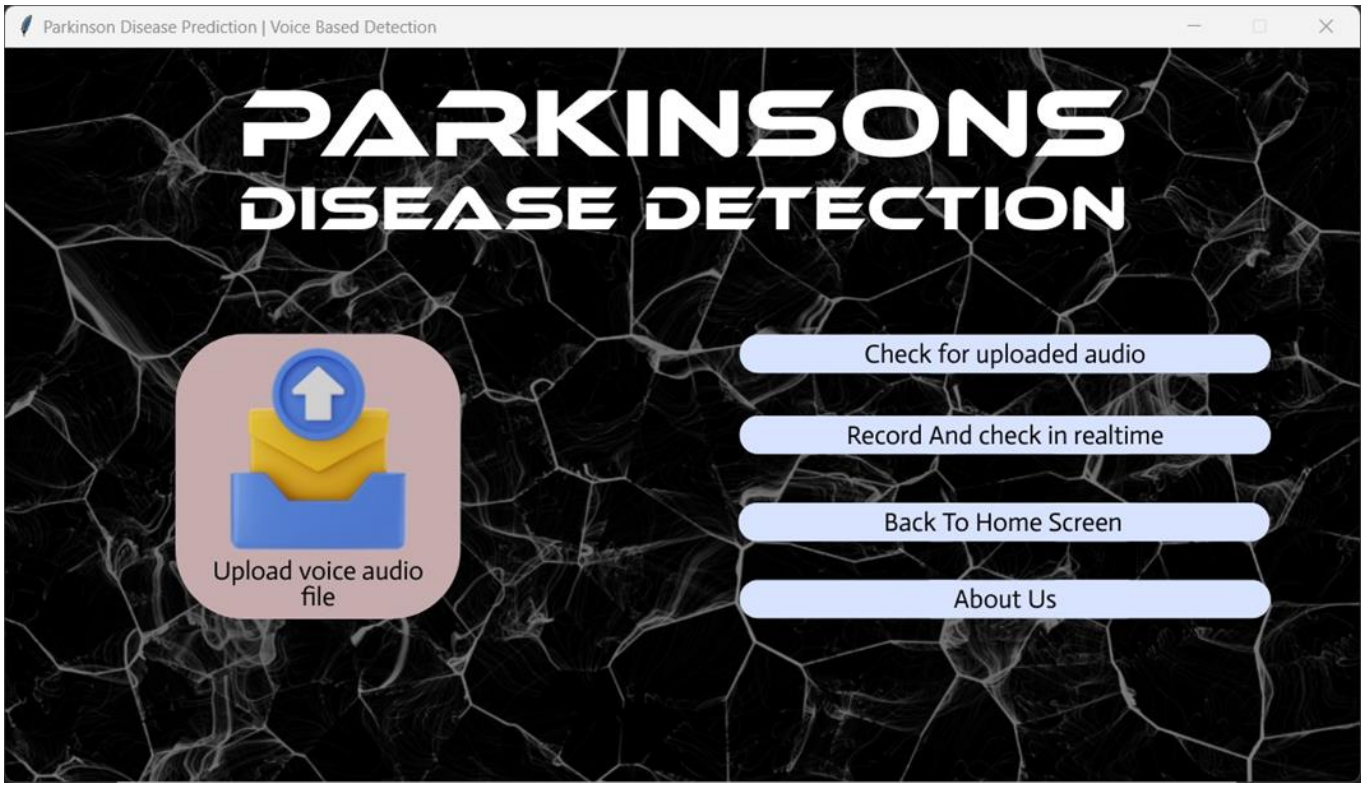


FIG 10: AUDIO BASED PREDICTION PAGE

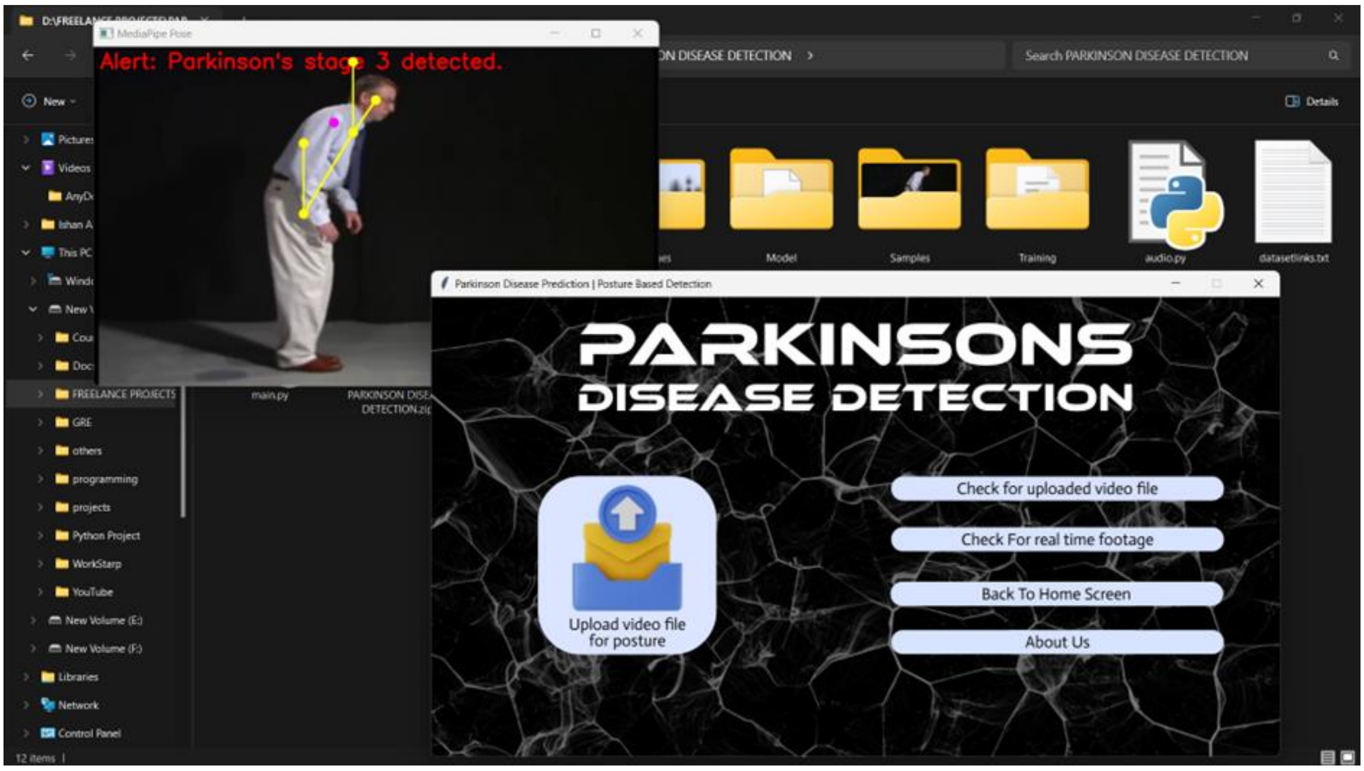
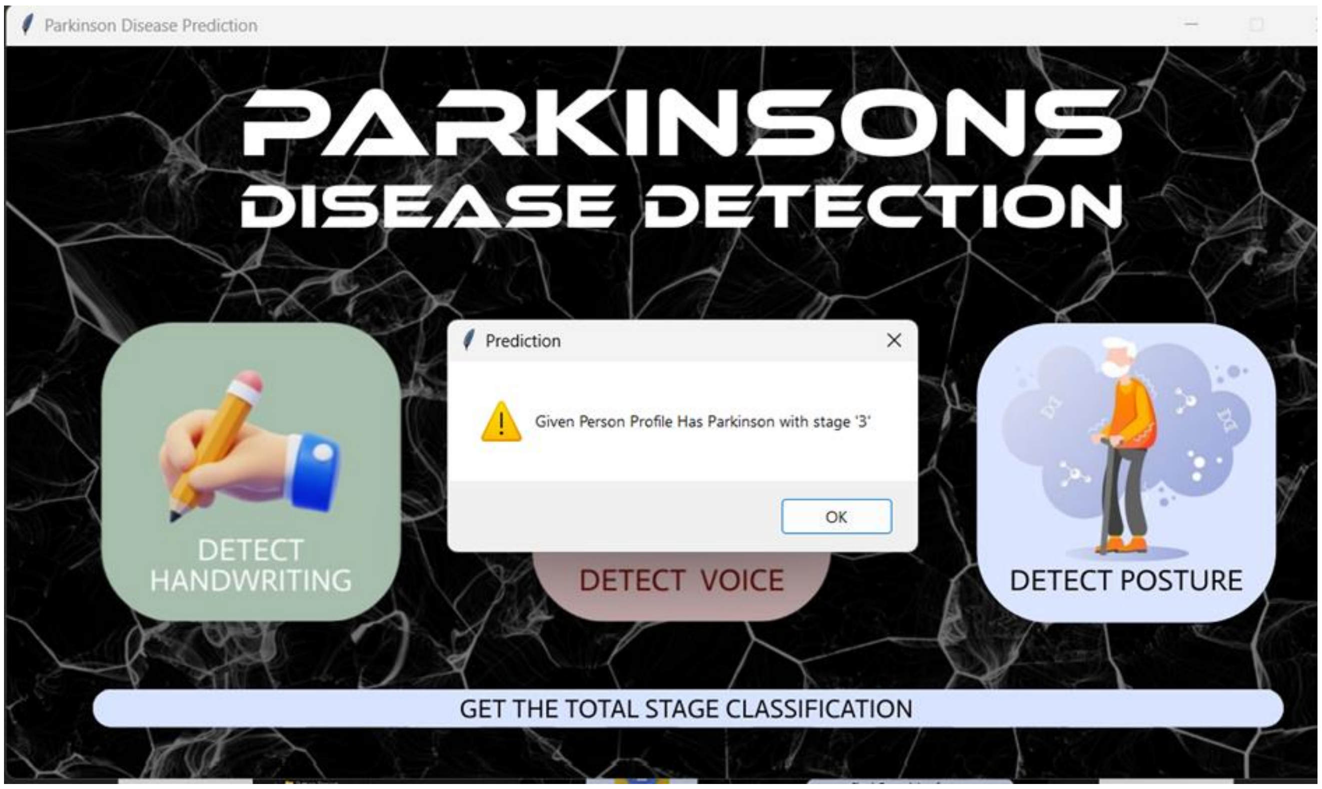
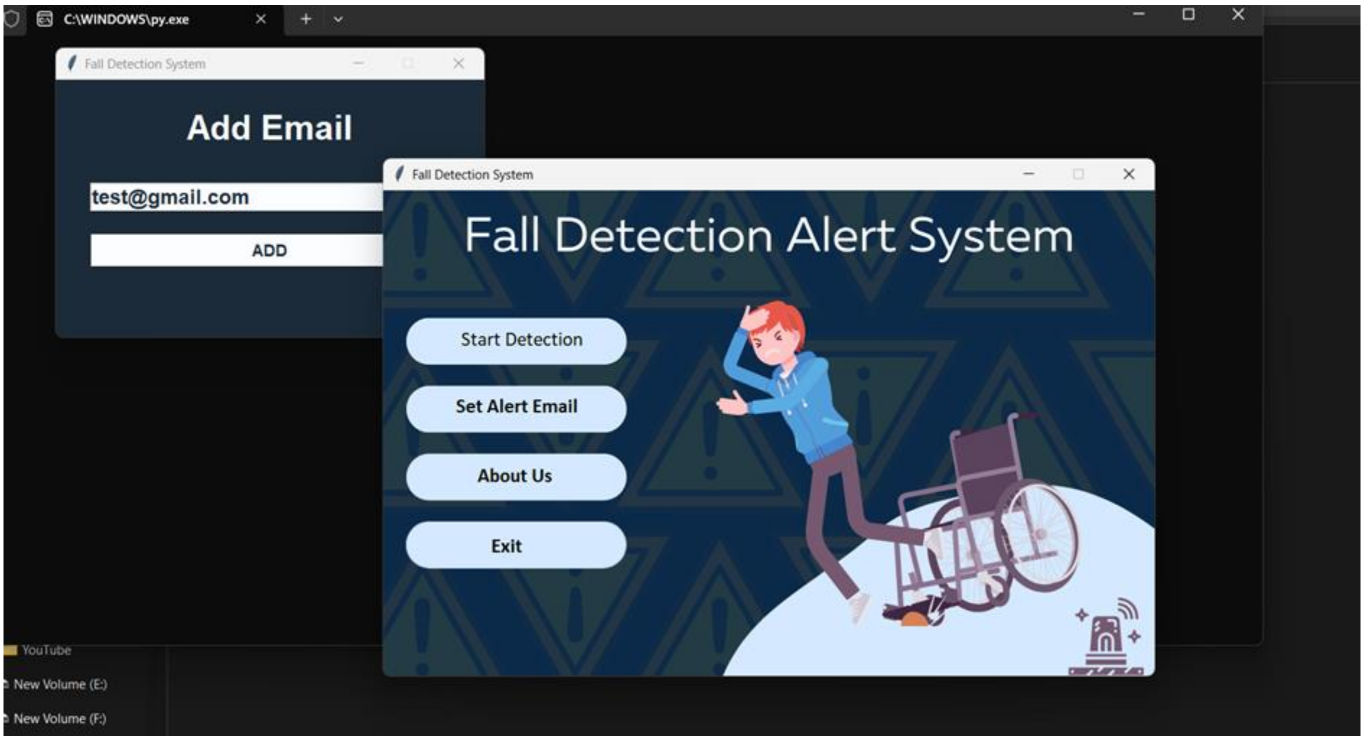

MediaPipe Pose
Alert: Parkinson's stage 3 detected.
Parkinson Disease Prediction | Posture Based Detection
PARKINSONS
DISEASE DETECTION
Upload video file for posture
Check for uploaded video file
Check For real time footage
Back To Home Screen
About Us

FIG 11: BODY POSE BASIS STAGE CLASSIFICATION PAGE



Given Person Profile Has Parkinson with stage '3'

FIG 12: OVERALL STAGE CLASSIFICATION



test@gmail.com

FIG 13: FALL DETECTION EMAIL SETTING

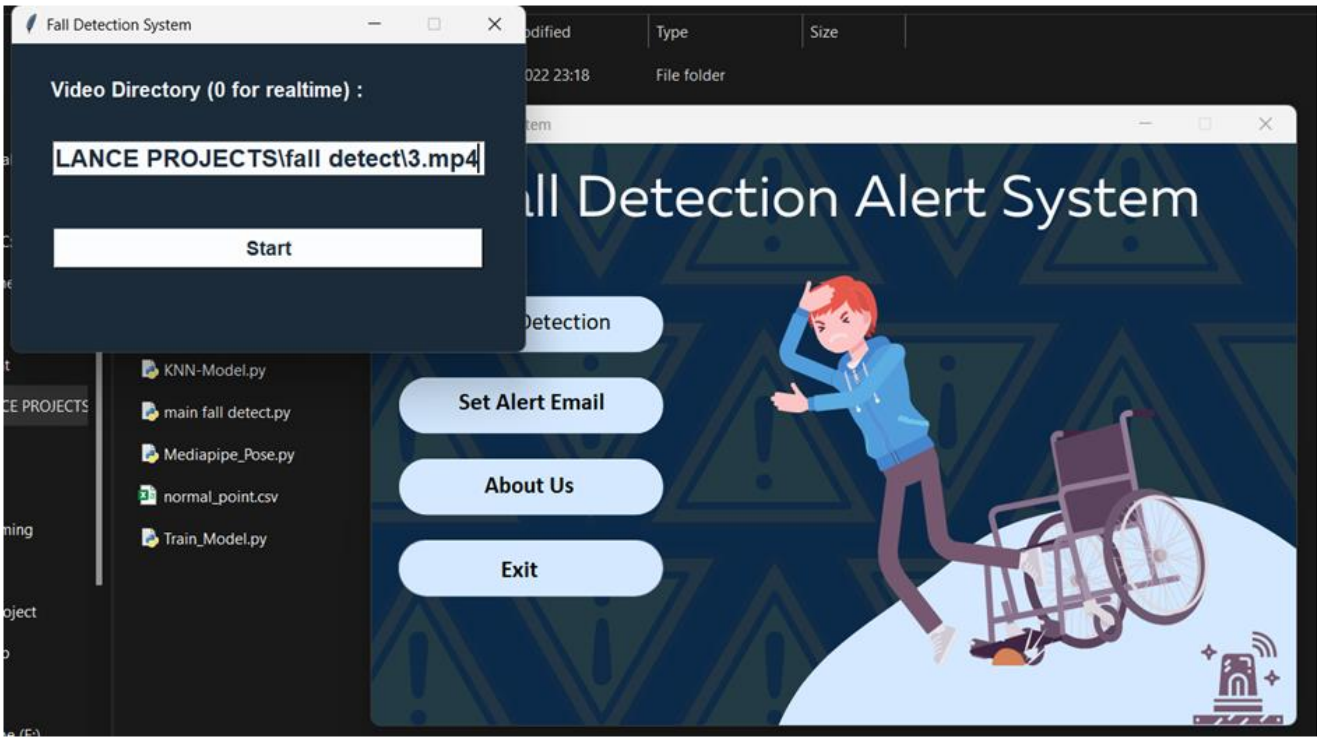


FIG 14: FALL DETECT VIDEO SETTING

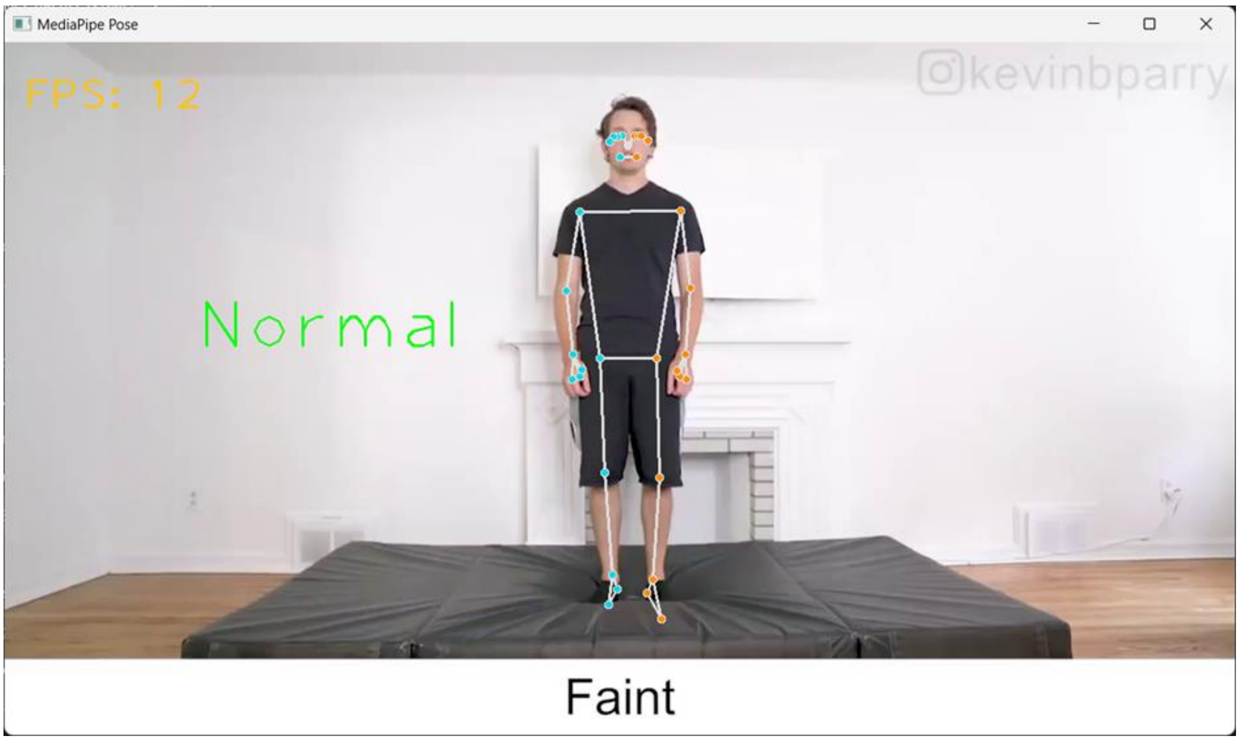


FIG 15: FALL DETECT POSE DETECTION

5. CONCLUSION

The pioneering Parkinson's disease (PD) management system introduced in this work heralds a monumental leap forward in addressing the multifaceted challenges that have beset the field for decades. Seamlessly integrating cutting-edge technologies, advanced data analysis techniques, and innovative methodologies, this system offers transformative solutions to some of the most pressing issues in PD care.

Historically, one of the most significant hurdles in PD management has been the inability to obtain objective, quantifiable, and consistent data for accurate disease monitoring and progression tracking. Traditional methods have heavily relied on subjective clinical observations and assessments, which are susceptible to inconsistencies, biases, and variability among healthcare professionals. This system revolutionizes data collection by harnessing the power of computer vision, machine learning, and advanced signal processing algorithms.

The innovative integration of Convolutional Neural Networks (CNNs) for handwriting analysis, XGBoost for voice input analysis, and MediaPipe for body movement analysis enables the extraction of precise, quantitative data from multiple modalities. This approach not only eliminates subjectivity but also provides a comprehensive view of the patient's condition, capturing both motor and non-motor symptoms with remarkable accuracy.

Another long-standing challenge has been the lack of continuous, real-time monitoring of PD patients, which often leads to delays in detecting changes in disease progression or identifying potential health risks. The proposed system tackles this issue head-on by incorporating real-time analysis and monitoring capabilities across all three modalities. This continuous monitoring aspect empowers healthcare professionals with timely and actionable insights, enabling them to make informed decisions and adjust treatment plans promptly.

Moreover, the system addresses the critical need for personalized care, which has been a significant gap in traditional PD management approaches. By leveraging the wealth of data collected from handwriting, voice, and body movement analysis, the system can identify unique patterns and characteristics specific to each individual. This personalized insight paves the way for tailored intervention strategies, optimizing therapeutic outcomes, and enhancing the overall quality of life for patients.

Recognizing the heightened risk of falls and associated complications in PD patients, the system incorporates a groundbreaking real-time fall detection mechanism. Utilizing advanced computer vision techniques and KNN algorithms, the system can swiftly detect falls and trigger immediate alerts to caregivers or emergency services. This innovative feature addresses the urgent need for enhanced patient safety, rapid response, and potential life-saving interventions.

Furthermore, the system overcomes the challenges of data inconsistency and subjectivity in PD evaluation by employing rigorous data preprocessing techniques and leveraging the power of machine learning algorithms. The use of CNN for handwriting analysis, XGBoost for voice input analysis, and MediaPipe for body movement analysis ensures that data is processed and analyzed in a consistent, objective, and reproducible manner, minimizing the risk of biases and variability.

By addressing these prominent and long-standing challenges, the proposed PD management system represents a paradigm shift in the field of neurology and healthcare. It not only offers innovative solutions but also paves the way for future advancements and breakthroughs. With its comprehensive approach, cutting-edge technologies, and commitment to personalized care, this system holds the potential to significantly improve the lives of individuals with Parkinson's disease, enhance the effectiveness of disease management strategies, and contribute to the broader goal of advancing patient-centric care.

6. REFERANCES

1. Pereira, C. R., Pereira, D. R., Weber, S. A., Hook, C., de Albuquerque, V. H. C., & Papa, J. P. (2016). Handwritten dynamics for Parkinson's disease identification. In 2016 International Joint Conference on Neural Networks (IJCNN) (pp. 4769-4776). IEEE.
2. Tsanas, A., Little, M. A., McSharry, P. E., & Ramig, L. O. (2010). Nonlinear speech analysis algorithms mapped to a standard metric for the measurement of Parkinson's disease. In 2010 International Conference on Acoustics, Speech and Signal Processing (pp. 1006-1009). IEEE.
3. Zhan, A., Little, M. A., Harris, D. A., Munoz, S. A., Ferris, N. J., Toth, E. E., ... & Wong, S. Q. (2020). High-dimensional longitudinal classification with pitfalls: An example on Parkinson's disease. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 28(1), 258-267
4. Gabel, M., Gilad-Bachrach, R., Renshaw, E., & Schuster, A. (2012). Full body gait analysis with Kinect. In 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (pp. 1964-1967). IEEE.
5. Capecci, M., Pepa, L., Verdini, F., & Ceravolo, M. G. (2016). A smartphone-based architecture to detect and quantify freezing of gait in Parkinson's disease. Gait & Posture, 50, 28-33.
6. Zhang, Z., Chuang, J., Huang, Y. L., Tsang, K. K., & Chan, S. C. (2021). Computer vision and machine learning for fall detection and alert systems: A survey. Sensors, 21(18), 6203.
7. Source websites : www.kaggle.com, www.wikipedia.com, www.stackoverflow.com, www.researchgate.py, www.github.com