AI BODY LANGUAGE DECODER USING MEDIAPIPE AND PYTHON

The Project Report Submitted in partial fulfilment of the requirement for the award of degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & SYSTEMS ENGINEERING

Submitted by

J SRI GAYATHRI SEELAMANTHULA	**317114110022**
MADHURIMA KALIVARAPU	**317114110023**
SAI PRATHYUSHA KANISETTI	**317114110024**
SANKEERTHANA RAJAN KAREM	**317114110025**

Under the esteemed guidance of

Dr. K. SOUMYA

Assistant Professor (C)

DEPARTMENT OF CS & SE, AUCEW.

DEPARTMENT OF COMPUTER SCIENCE & SYSTEMS ENGINEERING,

ANDHRA UNIVERSITY COLLEGE OF ENGINEERING FOR WOMEN,

ANDHRA UNIVERSITY, VISAKHAPATNAM-530017

JUNE, 2021.

CERTIFICATE

ANDHRA UNIVERSITY COLLEGE OF ENGINEERING FOR WOMEN
VISAKHAPATNAM

This is to certify that the project entitled "**AI Body Language Decoder using MediaPipe and Python**" is the bonafide work done by **J SRI GAYATHRI SEELAMANTHULA (317114110022), MADHURIMA KALIVARAPU (317114110023) , SAI PRATHYUSHA KANISETTI (317114110024) , SANKEERTHANA RAJAN KAREM (317114110025)** submitted in partial fulfilment of the requirement for the Award of Degree of Technology in Computer Science and Systems Engineering during the year 2020-2021.

Signature of the Project Guide
Dr. K. Soumya,
Assistant professor(c),
Department of CS & SE,
AUCEW.

Signature of Head of the Department
Prof. B. Prajna,
Head of the Department
Department of CS & SE,
AUCEW

DECLARATION

We hereby, declare that the project entitled **"AI Body Language Decoder using Mediapipe and Python"** is an authenticated record of our own work submitted by us in partial fulfilment of the requirement for the award of Degree of Bachelor of Technology in **COMPUTER SCIENCE AND SYSTEMS ENGINEERING** by **ANDHRA UNIVERSITY, VISAKHAPATNAM.**

317114110022 **J SRI GAYATHRI SEELAMANTHULA**

317114110023 **MADHURIMA KALIVARAPU**

317114110024 **SAI PRATHYUSHA KANISETTI**

317114110025 **SANKEERTHANA RAJAN KAREM**

ACKNOWLEDGEMENT

Apart from our efforts, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in successful completion of this project.

We would like to express our deep sense of gratitude and indebtedness to our esteemed institute "**ANDHRA UNIVERSITY COLLEGE OF ENGINEERING FOR WOMEN**", which has provided us an opportunity to fulfill the most cherished desire to reach our goal.

Firstly, we would like to thank our project guide **Dr. K. Soumya**, Assistant Professor(c) for supervision and support.

We would like to thank **Prof. B. Prajna**, Head of the Department for the Department of Computer Engineering, Andhra University College of Engineering for Women for her official support for the progress of our project.

We would like to show our gratitude to our beloved principal **Prof. S. K. Bhatti** for her encouragement in the completion of our course from time to time.

Our sincere thanks go to all the faculty members and non-teaching staff, Department of Computer Engineering for their supportive attitude.

Our thanks and appreciation also go to our parents and friends, who have helped us with their abilities in completing the project.

With gratitude,

J SRI GAYATHRI SEELAMANTHULA (317114110022)
MADHURIMA KALIVARAPU (317114110023)
SAI PRATHYUSHA KANISETTI (317114110024)
SANEERTHANA RAJAN KAREM (317114110025)

INDEX

ABSTRACT

Body language are visual languages produced by the movement of the hands, face and body. In this project we evaluate representations based on skeleton poses, as these are explainable, person-independent, privacy-preserving, low-Dimentional representations. Basically skeletal representations generalize over an individual's appearance and background, allowing us to focus on the recognition of motion. We present a real-time on-device body tracking pipeline that predicts hand skeleton and the whole body notion. It is implemented via MediaPipe, a framework for building cross-platform ML solutions. We perform using pose estimation systems and analyze the applicability of the estimation systems to body language recognition by evaluating failure cases of the existing models. The proposed system and architecture demonstrates real-time inference and high prediction quality.

LIST OF FIGURES

LIST OF TABLES

CHAPTER – 1

INTRODUCTION

1. INTRODUCTION`

Body Language Decoder helps detect and predict facial expressions, hand gestures and body pose. Facial expression recognition can help market research companies scale data and analyze quickly. The ability to perceive the shape and motion of hands can be a vital component in improving the user experience across a variety of technological domains and platforms. For example, it can form the basis for sign language understanding and hand gesture control, and can also enable the overlay of digital content and information on top of the physical world in augmented reality.[1]

Applications:

- ➢ Driver drowsiness detection.
- ➢ Sign language detection.
- ➢ Market research companies can use this technology to analyze data.
- ➢ Body language detection in interviews.

Pipeline Design

The ML pipeline makes use of multiple independent models, each of which is locally-optimized to their task — pose, hand, and face detection. While they coordinate with each other, they use separate inputs, and the input vector for one model may not be suitable for another model.

For example, the pose detection model makes use of a lower resolution (256x256) video frame as its input. However, if the regions for the face or hands were to be cropped from this frame, the resulting frame would have a resolution that's much too low to be used accurately as an input for the other models. As a result, each model makes use of its own input frame from the original captured video feed.

The outcome, thus, is a multi-stage pipeline, where each model gets its own frame that zeroes in on the corresponding region of interest (face, hand, etc.), with the resolution that's deemed appropriate for that specific region.

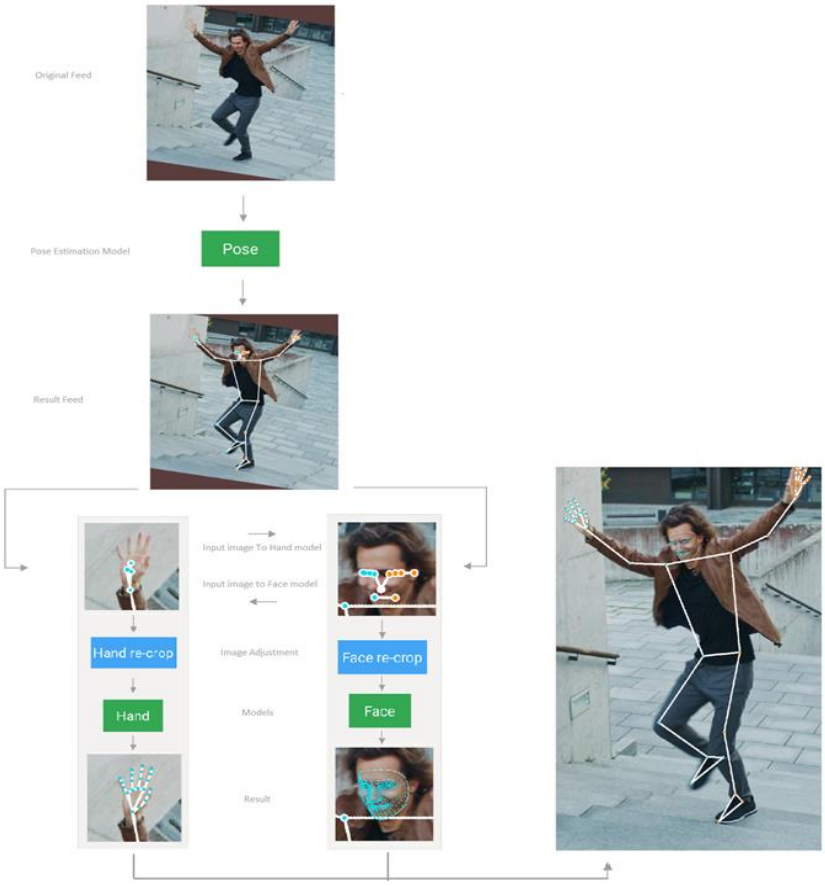


Figure 1.1 Pipeline Design

The pose estimation model thus comes first in the pipeline. MediaPipe makes use of the BlazePose pose estimation model for this. To maximize performance, the pipeline assumes that the object does not move significantly from frame-to-frame, so the result of the previous frame—i.e., the body region of interest—can be used to start inference on a new frame.

Landmark Detection[2]

This project uses Mediapipe's holistic landmark subgraph. The holistic landmark subgraph uses three separate models internally:

a) The pose landmark module

The MediaPipe Pose Landmark model allows for high-fidelity body pose tracking, inferring 33 2D landmarks on the whole body (or 25 upper-body landmarks) from RGB

video frames, utilizing BlazePose. It detects the landmarks of a single body pose, full-body by default, but it can be configured to cover the upper-body only, in which case it only predicts the first 25 landmarks. It can run using either CPU or GPU depending on the type of module chosen.

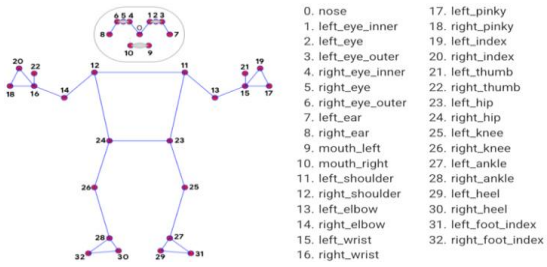


Figure 1.2 Pose Landmark

b) The hand landmark module

MediaPipe Hands is a high-fidelity hand and finger tracking solution. It can infer up to 21 3D landmarks of a hand from just a single frame. It's a hybrid between a palm detection model that operates on the full image and returns an oriented hand bounding box and a hand landmark model that operates on the cropped image region defined by the palm detector, which returns high-fidelity 3D hand keypoints. It detects landmarks of a single hand or multiple hand depending on the module type. The option to run using CPU or GPU is also featured.

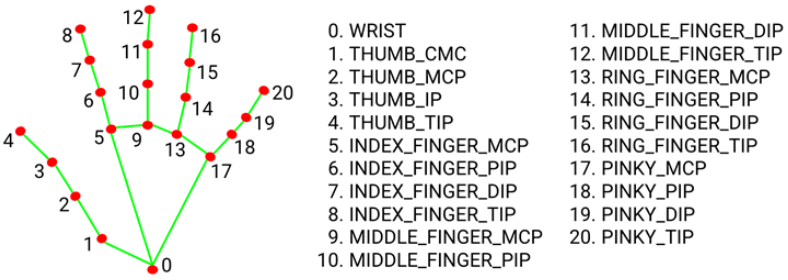


Figure 1.3 Hand Landmark

c) The face landmark module

The MediaPipe Face Mesh estimates 468 3D face landmarks in real-time on mobile devices. It employs deep neural networks to infer the 3D surface geometry, requiring only a single camera input, without the need for a dedicated depth sensor. Utilizing lightweight model

architectures with optional GPU acceleration throughout the pipeline, it can track landmarks on a single face or multiple faces. Additionally, it establishes a metric 3D space and uses the face landmark screen positions to estimate face geometry within that space.

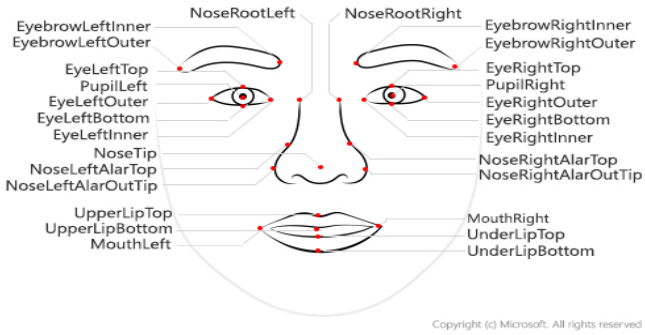


Figure 1.4 Face Landmark

1.1 PROBLEM STATEMENT

Sentiment Analysis is already widely used by different companies to gauge consumer mood towards their product or brand in the digital world. However, in the offline world users are also interacting with the brands and products in retail stores, showrooms, etc. and solutions to measure user's reaction automatically under such settings has remained a challenging task. Emotion Detection from facial expressions using AI can be a viable alternative to automatically measure consumer's engagement with their content and brands. On the other hand detecting body language, which is considered as one of the most effective communication method, can help interviewers analyze the candidate during the process.

This can be achieved by creating an excel sheet of coordinate points of landmarks of desired poses and training the model on that. This trained model can then be stored and used later for predicting user's gestures.

1.2 PROJECT DELIVERABLES

- ➢ Project Information
- ➢ Project Documentation
- ➢ Proposed System
- ➢ Requirements List

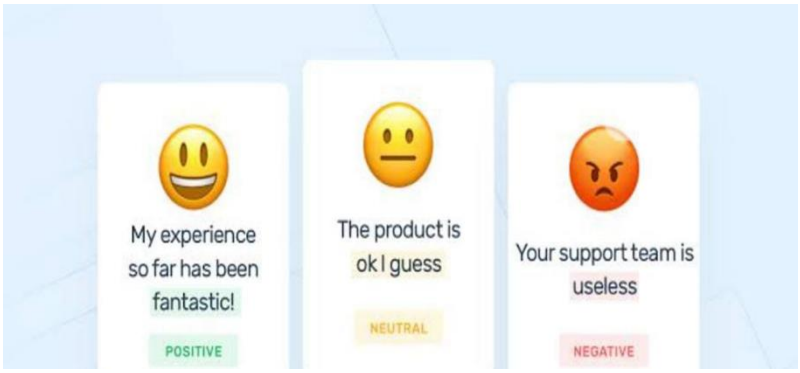
➢ Program

1.3 LIMITATIONS OF EXISTING TECHINIQUES

The popular models are the face expression detection to recognize the emotion and the text emotion widely used on social networking sites, the new sign detection added where it takes the hand gestures using hand landmarks and detects what kind of sign is made. The biometric attendance system is one the most used model in every sector where the face landmarks are stored. Given examples-

```
modi wil not win this time
[0.0471645]
Negative
```

The emotion of the sentence is detected



The emotion of the sentence with facial expressions

Figure 1.5 Sentence emotion

CONS –

- The complete body posture cannot be detected
- No real-time on- device tracking
- No accurate estimation
- In few existing methods, to predict the body language there need to white background which is a drawback.

1.4 PROPOSED SYSTEM

This system gives the recognition to the complete body/shape/ surface. From Fig 1.6, Using holistic MediaPipe the face, hand and as well as body posture is detected. It collects the coordinates and merges them giving the required output.

- Face detection
- Body gesture
- Pose estimation
- Real time Hand tracking

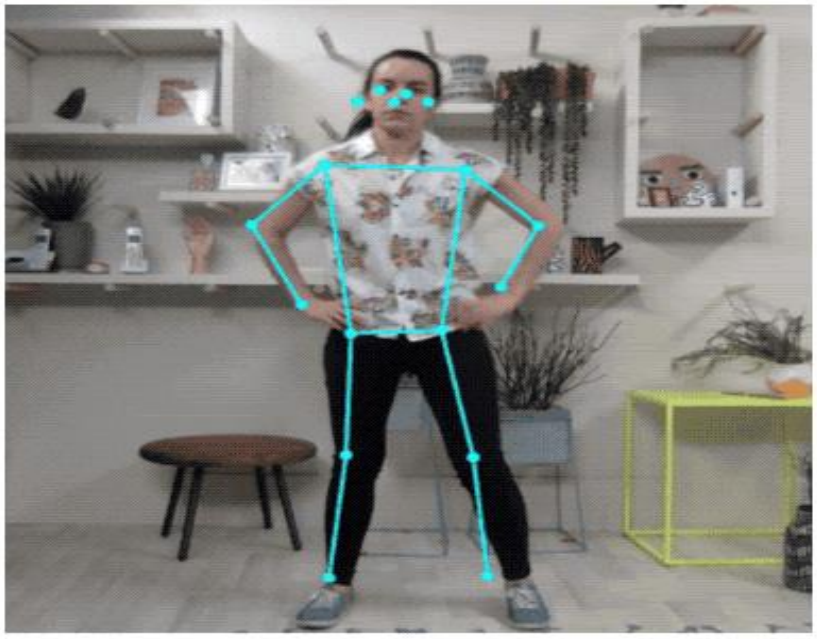


Figure 1.6 Holistic Mediapipe [3]

CHAPTER – 2
REVIEW OF LITERATURE

2. LITERATURE REVIEW

This chapter presents the research papers studied to carry out this project work

2.1 LITERATURE REVIEW ON BODY LANGUAGE DETECTION

Danilo Avola, Luigi Cinque, Stefano Levialdi, Giuseppe Placidi, in their paper "Human Body Language Analysis: A Preliminary Study Based on Kinect Skeleton Tracking", The nonverbal communication can be informally defined as the communicative process between two or more entities (e.g., persons) which achieving an informative exchange without using the semantic meaning of the words. This process can be accomplished by using one or more language forms, including the body language (i.e., movements, gestures, and postures) which in turn can be composed by voluntary and involuntary behaviours. The analysis and interpretation of these behaviours can infer different internal states of persons (e.g., feelings, attitudes, emotions) which in turn can support the development of a wide range of automatic applications in different fields, such as: rehabilitation, security, people identification, human behaviour analysis, biometric.

The purpose of the framework is to provide a tool to analyze and interpret verbal and nonverbal human-to-human communication in order to transfer this ability to the human-machine interaction.[4]

2.2 LITERATURE REVIEW ON MEDIAPIPE

Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg and Matthias Grundmann, in their paper "MediaPipe: A Framework for Building Perception Pipelines", MediaPipe is a framework for building pipelines to perform inference over arbitrary sensory data. With MediaPipe, a perception pipeline can be built as a graph of modular components, including model inference, media processing algorithms and data transformations, etc. Sensory data such as audio and video streams enter the graph, and perceived descriptions such as object-localization and face landmark streams exit the graph. An example is shown in Figure 1. MediaPipe is designed for machine learning (ML) practitioners, including researchers, students, and software developers, who implement production-ready ML applications, publish code

accompanying research work, and build technology prototypes. The main use case for MediaPipe is rapid prototyping of perception pipelines with inference models and other reusable components. MediaPipe also facilitates the deployment of perception technology into de mos and applications on a wide variety of different hardware platforms. MediaPipe enables incremental improvements to perception pipelines through its rich configuration language and evaluation tools. MediaPipe addresses these challenges by abstracting and connecting individual perception models into maintainable pipelines. All of the steps necessary to infer from the sensory data and get the perceived results are specified in the pipeline configuration. It is easy to re-use MediaPipe components in different pipelines across successive applications as these components share a common interface oriented around time-series data. Each pipeline can then run with the same behavior on a variety of platforms, enabling the practitioner to develop the application on workstations, and then deploy it on mobile, for example. MediaPipe consists of three main parts: (a) a framework for inference from sensory data, (b) a set of tools for performance evaluation, and (c) a collection of re-usable inference and processing components called calculators.[5]

2.3 LITERATURE REVIEW ON SCIKIT-LEARN

In Journal of Machine Learning research, “Scikit-learn: Machine Learning in Python”, The Python programming language is establishing itself as one of the most popular languages for scientific computing. Thanks to its high-level interactive nature and its maturing ecosystem of scientific libraries, it is an appealing choice for algorithmic development and exploratory data analysis (Dubois, 2007; Milmann and Avaizis, 2011). Yet, as a general-purpose language, it is increasingly used not only in academic settings but also in industry. Scikit-learn harnesses this rich environment to provide state-of-the-art implementations of many well known machine learning algorithms, while maintaining an easy-to-use interface tightly integrated with the Python language. This answers the growing need for statistical data analysis by non-specialists in the software and web industries, as well as in fields outside of computer-science, such as biology or physics. Scikit-learn differs from other machine learning toolboxes in Python for various reasons: i) it is distributed under the BSD license ii) it incorporates compiled code for efficiency, unlike MDP (Zito et al., 2008) and pybrain (Schaul et al., 2010), iii) it depends only on numpy and scipy to facilitate easy distribution, unlike pymvpa (Hanke et al., 2009) that has optional dependencies such as R and

shogun, and iv) it focuses on imperative programming, unlike pybrain which uses a data-flow framework. While the package is mostly written in Python, it incorporates the C++ libraries LibSVM (Chang and Lin, 2001) and LibLinear (Fan et al., 2008) that provide reference implementations of SVMs and generalized linear models with compatible licenses Binary packages are available on a rich set of platforms including Windows and any POSIX platforms. 2826 SCIKIT-LEARN: MACHINE LEARNING IN PYTHON Furthermore, thanks to its liberal license, it has been widely distributed as part of major free software distributions such as Ubuntu, Debian, Mandriva, NetBSD and Macports and in commercial distributions such as the “Enthought Python Distribution”.[6]

CHAPTER – 3
SYSTEM DESIGN

3. SYSTEM DESIGN

3.1 Introduction

A Human Pose Skeleton represents the orientation of a person in a graphical format. Essentially, it is a set of coordinates that can be connected to describe the pose of the person. Each coordinate in the skeleton is known as a part (or a joint, or a key point). A valid connection between two parts is known as a pair (or a limb). Note that, not all part combinations give rise to valid pairs. A sample human pose skeleton is shown.

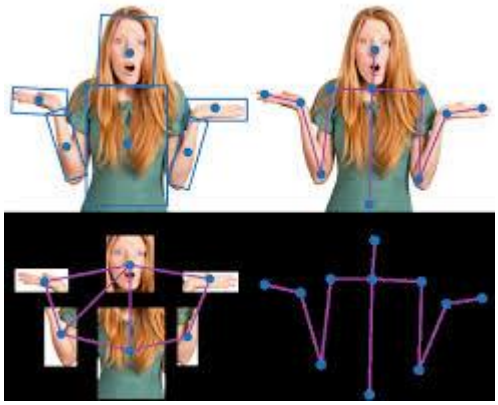


Figure 3.1 Human Pose skeleton, with coordinate points

Knowing the orientation of a person opens avenues for several real-life applications, some of which are discussed towards the end of this blog. Several approaches to Human Pose Estimation were introduced over the years. The earliest (and slowest) methods typically estimating the pose of a single person in an image which only had one person to begin with. These methods often identify the individual parts first, followed by forming connections between them to create the pose.

Naturally, these methods are not particularly useful in many real-life scenarios where images contain multiple people.

3.2 UML Diagrams

3.2.1 Use Case Diagram

Use cases represent only the functional requirements of a system. Other requirements such as business rules, quality of service requirements, and implementation constraints must be represented separately, again, with other UML diagrams.[7]

Use case diagrams are typically developed in the early stage of development and people often apply use case modelling for the following purposes:

- Specify the context of a system.
- Capture the requirements of a system.
- Validate a systems architecture.
- Drive implementation and generate test cases.
- Developed by analysts together with domain experts.

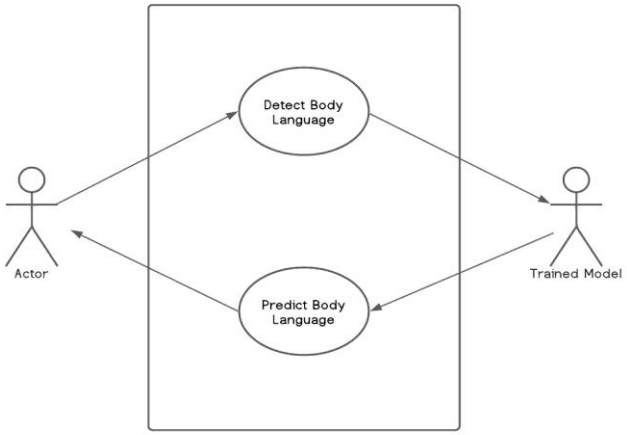


Figure 3.2 Use Case Diagram

3.2.2 State Diagram

A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied. A state transition is a relationship between two states, two activities, or between an activity and a state.

We can show one or more state transitions from a state as long as each transition is unique. Transitions originating from a state cannot have the same event, unless there are conditions onthe event.

Use cases and scenarios provide a way to describe system behavior; in the form of

interaction between objects in the system. Sometime it is necessary to consider inside behavior of an object.

A state chart diagram shows s the states of a single objects, the events or messages that cause a transition from one state to another and the actions that result from a state change. As I activity diagram, state chart diagram also contains special symbols for start state and stop state.

State chart diagram cannot be created for every class in the system, it is only for those lass objects with significant behavior.

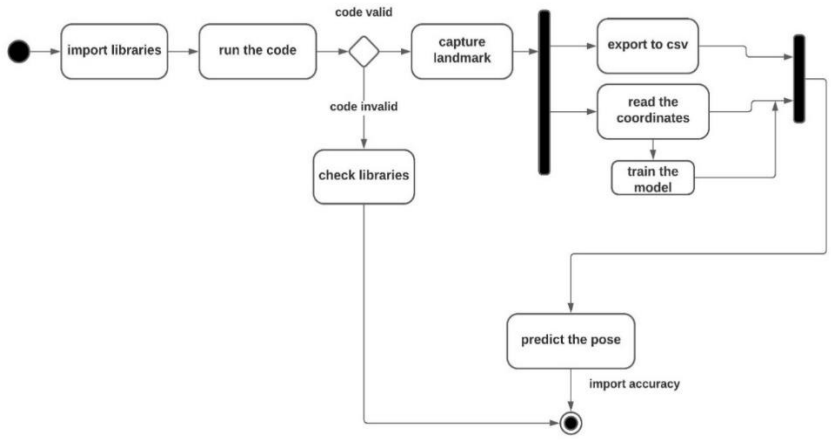


Figure 3.3 State Diagram

3.2.3 Activity Diagram

An Activity diagram is a variation of a special case of a state machine, in which the states are activities representing the performance of operations and the transitions are triggered by the completion of the operations.The purpose of Activity diagram is to provide a view of flows and what is going on inside a use case or among several classes.You can also use activity diagrams to model code-specific information such as a class operation.
Activity diagrams are very similar to a flowchart because you can model a workflow from activity to activity. An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities.

Activity diagrams contain activities, transitions between the activities, decision points, and synchronization bars. An activity represents the performance of some behavior in the workflow.

In the UML, activities are represented as rectangles with rounded edges, transitions are drawn as directed arrows, decision points are shown as diamonds, and synchronization bars are drawn as thick horizontal or vertical bars.

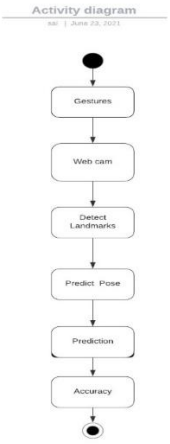


Figure 3.4 Activity Diagram

3.2.4 Object Diagram

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.

Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

Object diagrams are used to render a set of objects and their relationships as an instance.

The purpose of the object diagram can be summarized as −

- Forward and reverse engineering.
- Object relationships of a system
- Static view of an interaction.

- Understand object behaviour and their relationship from practical perspective

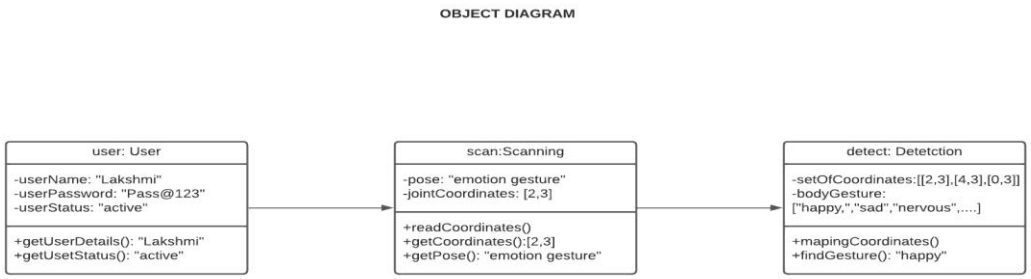


Figure 3.5 Object Diagram

3.2.5 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

The purpose of the class diagram can be summarized as −

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

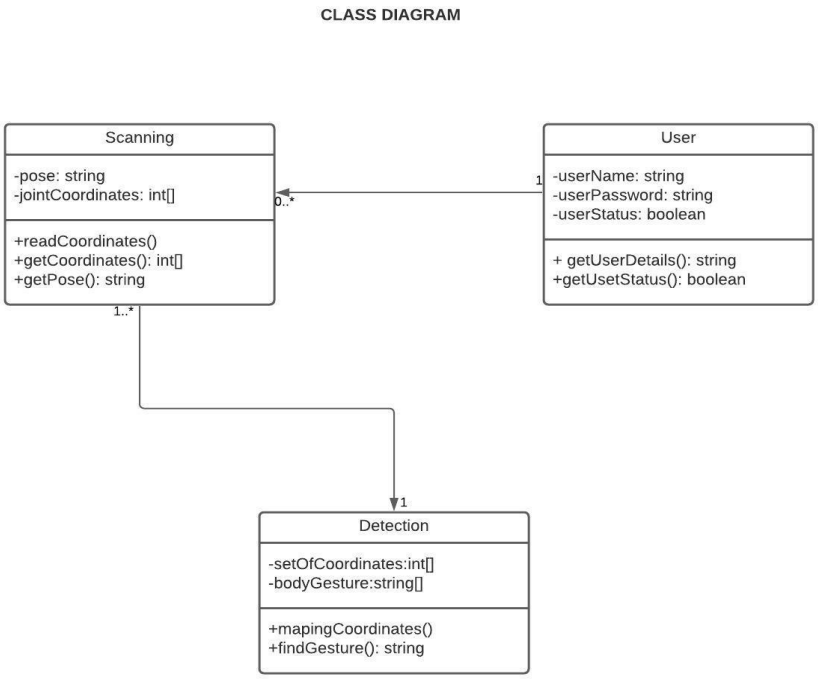


Figure 3.6 Class Diagram

3.2.6 Deployment Diagram

The second type of implementation diagram provided by UML is the deployment diagram. Deployment diagrams are used to show the configuration of run-time pro- cessing elements and the software components and processes that are located on them.

Deployment diagrams are made up of nodes and communication associations. Nodes are typically used to show computers and the communication associations show the network and protocols that are used to communicate between nodes. Nodes can be used to show otherprocessing resources such as people or mechanical resources.

Nodes are drawn as 3D views of cubes or rectangular prisms, and the following figure shows a simplest deployment diagram where the nodes connected by communication

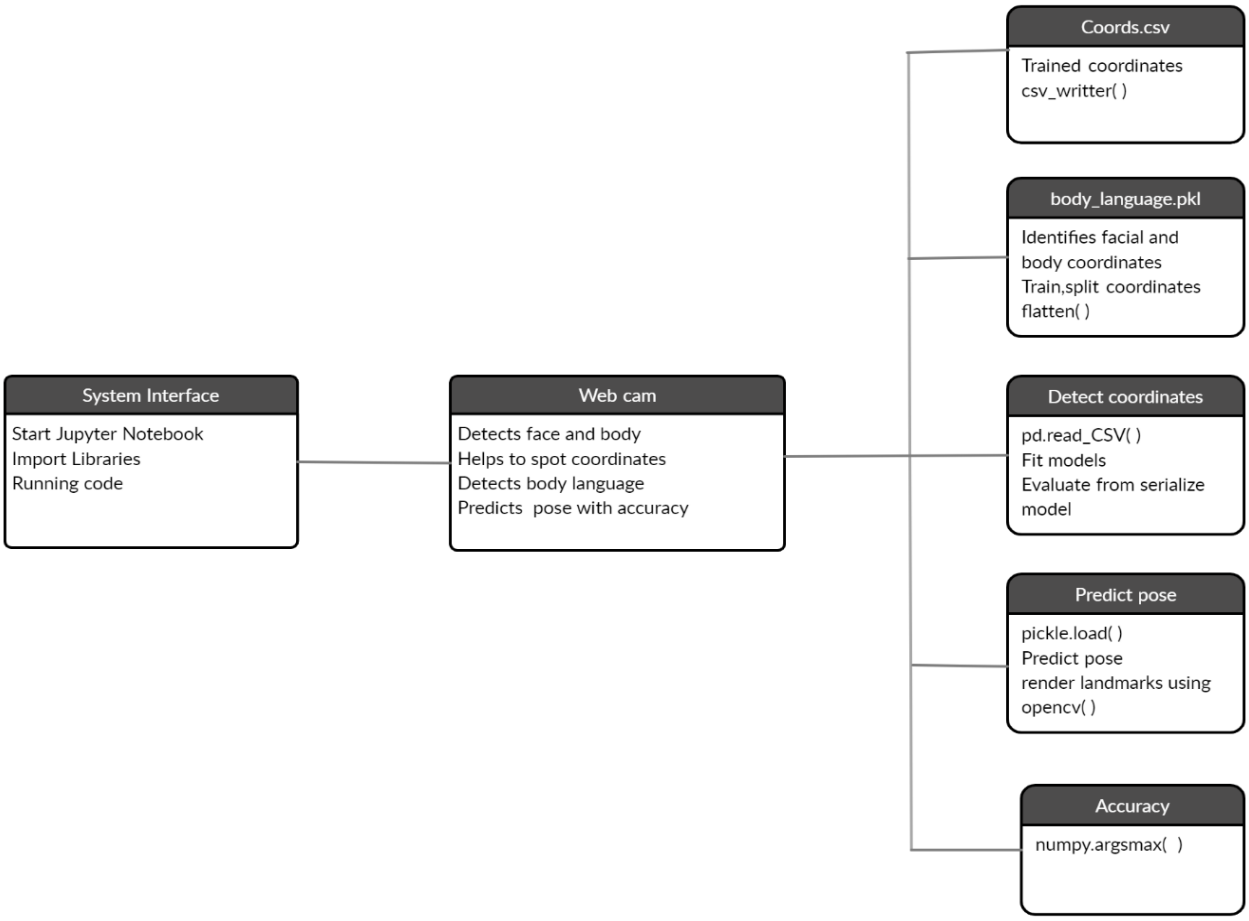


Figure 3.7 Deployment Diagram

3.2.7 Sequence Diagram

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence-what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

A sequence diagram has two dimensions: the vertical dimension represents time; the horizontal dimension represents different objects. The vertical line is called the object's lifeline. The lifeline represents the object's existence during the interaction.

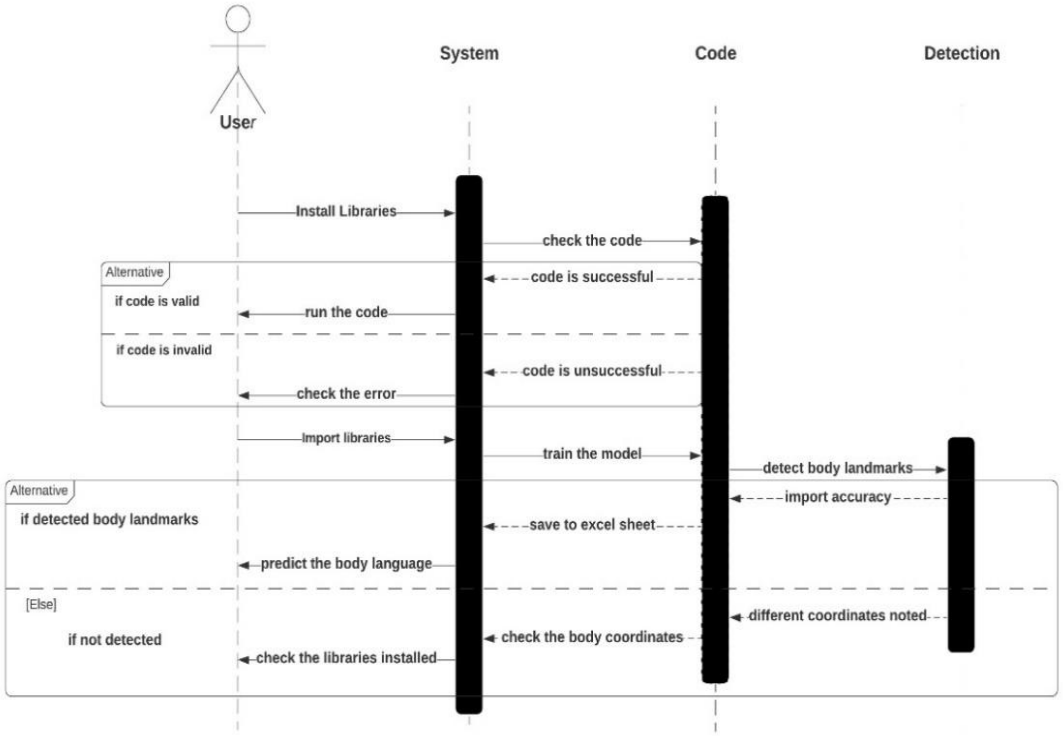


Figure 3.8 Sequence Diagram

3.2.8 Component Diagram

Component Diagrams show the dependencies between software components in the system. The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time or at run time.

In a large project there will be many files that make up the system. These files will have dependencies on one another. The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time, at link- time or at run-time. There are also dependencies between source code files and the executable files or byte code files that are derived from them by compilation. Component diagrams are one of the two types of implementation diagram in UML. Component diagrams show these dependencies between software components in the system. Stereotypes can be used to show

dependencies that are specific to particular languages also.

A component diagram shows the allocation of classes and objects to components in the physical design of a system. A component diagram may represent all or part of the component architectureof a system along with dependency relationships. The dependency relationship indicates that one entity in a component diagram uses the services or facilities of another.

Dependencies in the component diagram represent compilation dependencies.

- The dependency relationship may also be used to show calling dependencies among components, using dependency arrows from components to interfaces on other components. Different authors use component diagrams in different ways. Here we have the following distinction b/n them
- Components in a component diagram should be the physical components of a system.
- During analysis and the early stages of design, package can be used to show the logical grouping of class diagrams or of models that use other kinds of diagrams into packages relatingto sub-systems.
- During implementation, package diagrams can be used to show the grouping of physical components into sub-systems.

If component diagrams are used, it is better to keep separate sets of diagrams to show compile- time and run-time dependencies, however, this is likely to result in a large number of diagrams. Component diagrams show the components as types. If you wish to show instances components of components you can use a deployment diagram.

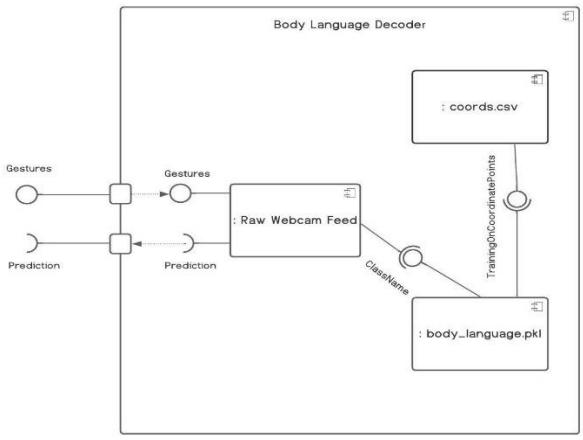


Figure 3.9 Component Diagram

3.3 SYSTEM FLOWCHART

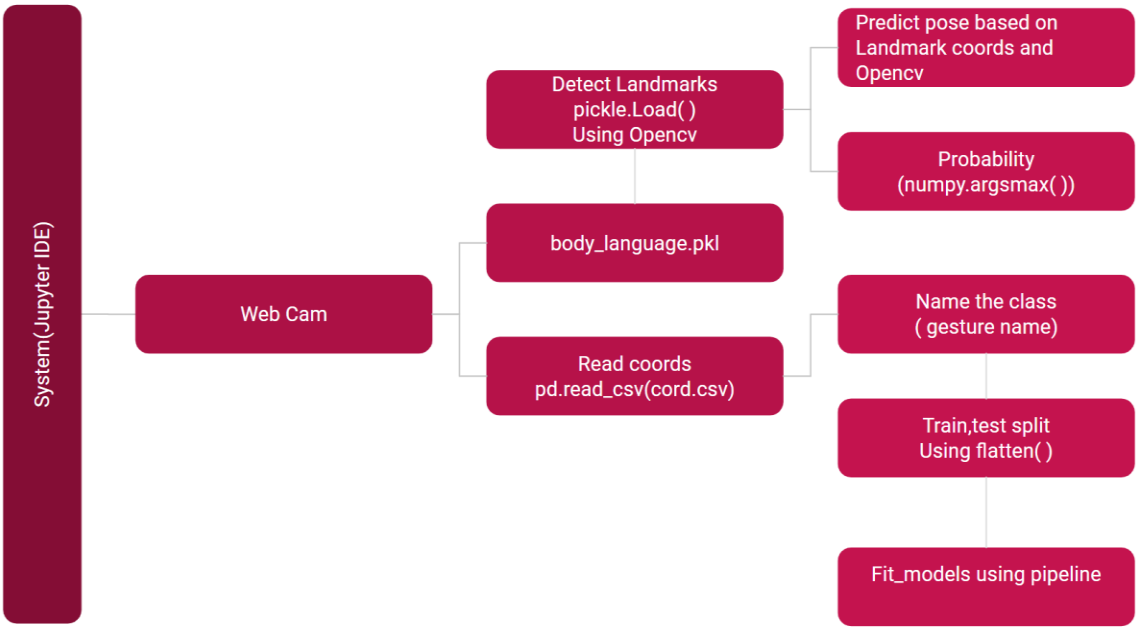


Figure 3.10 System FlowChart

Initially we run code on jupyter IDE. when code is executed, camera starts running. It reads the coordinates using holistic MEDIAPIPE and then compares it with predefined body_language.pkl

file when we trained earlier. Then it predicts the gesture. Also it predicts accuracy of the pose comparing to pre-trained gesture using numpy-argsmax( ) function imported from Numpy library.

In case of training the model, we run the code and web cam starts running. It reads the co-ordinates from our body using holistic function imported from MediaPipe library and writes to coords.csv file. We name the class as the name of the gesture we are training. These coordinates are then clustered using flatten( ) function imported from scikit- learning ML library.

CHAPTER – 4
SYSTEM ANALYSIS

4.SYSTEM ANALYSIS

Systems analysis is a problem solving technique that decomposes a system into its component pieces for the purpose of studying how well those component parts work and interact to accomplish their purpose. System analysis is the process of studying a procedure in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way.

The development of a computer-based information system includes a systems analysis phase which produces or enhances the data-model which itself is a precursor to creating or enhancing a database. There are a number of different approaches to system analysis. When a computer-based information system is developed, systems analysis would constitute the following steps:

- The development of a feasibility study, involving determining whether a project is economically, socially, technologically and organizationally feasible
- Conducting fact-finding measures, designed to ascertain the requirements of the system's end-users. These typically span interviews, questionnaires, or visual observations of work on the existing system.
- Gauging how the end-users would operate the system (in terms of general experience in using computer hardware or software), what the system would be used for and so on.

4.1 System Requirement Specification:

System requirements specification is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do. software requirements specifications permit a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.[8]

4.1.1 Functional Requirements:

Functional requirements describe the interactions between the system and its environment independent of its implementation. The environment includes the user and any other external system with which the system interacts. Functional requirements capture the intended to behavior of the system, this behavior may be expressed as services, tasks or functions the system is required to perform.

The system should be able to meet the following functionalities :

Input:

- Gestures through webcam.

Output:

- Gesture prediction along with probability

4.1.2 Non-Functional Requirement:

Non-functional requirements describe the aspects of the system that are not directly related to the functional behavior of the system. Non-functional requirements include a broad variety of requirements that apply to many different aspects of the system, from usability to performance.[9]

- **Portability**: The degree of conversion of our application to target system is easy (e.g., Windows, Unix etc.).
- **Efficiency:** Our application uses the CPU cycles, memory and disk space efficiently.
- **Understandability**: The UI is easily understandable by everyone.
- **Accuracy**: Our application gives accurate results what clients expect.
- **Robustness**: As we are using OPENCV in our application can handle any situation.
- **Usability**: It is very easy to learn and operate the system.
- **Cost and development time**: The cost and development time is very less.

4.2 Feasibility Study:

The feasibility study is an evaluation and analysis of the potential of a proposed project. It is based on extensive investigation and research to support the process of decision making. Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of an existing or proposed system, opportunities and threats present in the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained. A well-designed feasibility study should provide a historical background of a project, a description of a service, and details of the operations. Generally, feasibility studies precede technical development and project implementation. A feasibility study evaluates the project's potential for success. It must therefore be conducted with an objective, unbiased approach to provide information upon which decisions can be based.[10]

Scalability: Ability to process huge amounts of data.

Reliability: It is an efficient way of processing data without loss.

Technical Feasibility:

Technical feasibility also involves evaluation of the hardware and the software requirements of the proposed system.

Economic Feasibility:

It serves as an independent project assessment, and enhances project credibility. Economic feasibility is an assessment which typically involves cost/ benefits from the analysis of the project.

Operational Feasibility:

It measures how well the proposed system solves problems and takes advantage of the opportunities identified during scope definition.

Operational feasibility studies analyse how the project plan satisfies the requirements identified in the requirements analysis phase of system development. To ensure success, desired operational outcomes must inform and guide design and development. These include such design-dependent parameters such as reliability, maintainability, supportability, usability, disposability, sustainability, affordability and others.

Scheduling Feasibility:

It is an important factor for project success. A project will fail if it is not completed on time. In Scheduling feasibility, we estimate how much time the system will take to complete and with our technical skills we need to estimate the period to complete the project using various methods of estimation.

Benefits of Conducting a Feasibility Study:

Conducting a feasibility study is always beneficial to the project as it gives you and other stakeholders a clear picture of your idea. Below are the key benefits of conducting a feasibility study:

Gives project teams more focus and provides an alternative outline

- ❖ Narrows the business alternatives.
- ❖ Identifies a valid reason to undertake the project .
- ❖ Enhances the success rate by evaluating multiple parameters.
- ❖ Aids decision-making on the project.

Object-Oriented Analysis:

Object Oriented Analysis is a popular technical approach for analyzing, designing an application, system or business by applying the object-oriented paradigm and visual modelling throughout the development lifecycle to faster, better, stakeholder communication and product quality.

In the case of object-oriented analysis, the process varies. But these two are identical at use case analysis. Actually, the steps involved in the analysis phase are :

- ❖ Identify the actors.

- ❖ Classification-develops a static UML class diagram.
- ❖ Develop use cases.
- ❖ Identify classes, relationships, attributes, methods

4.3 Use Case Scenarios:

Use Case Model:

A Use case is a description of the behaviour of the system. That description is written from the point of a user who just told the system to do something particular.

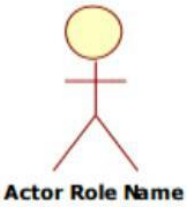
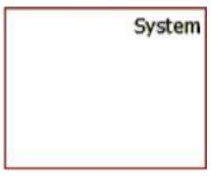
4.3.1 Use case Diagram:

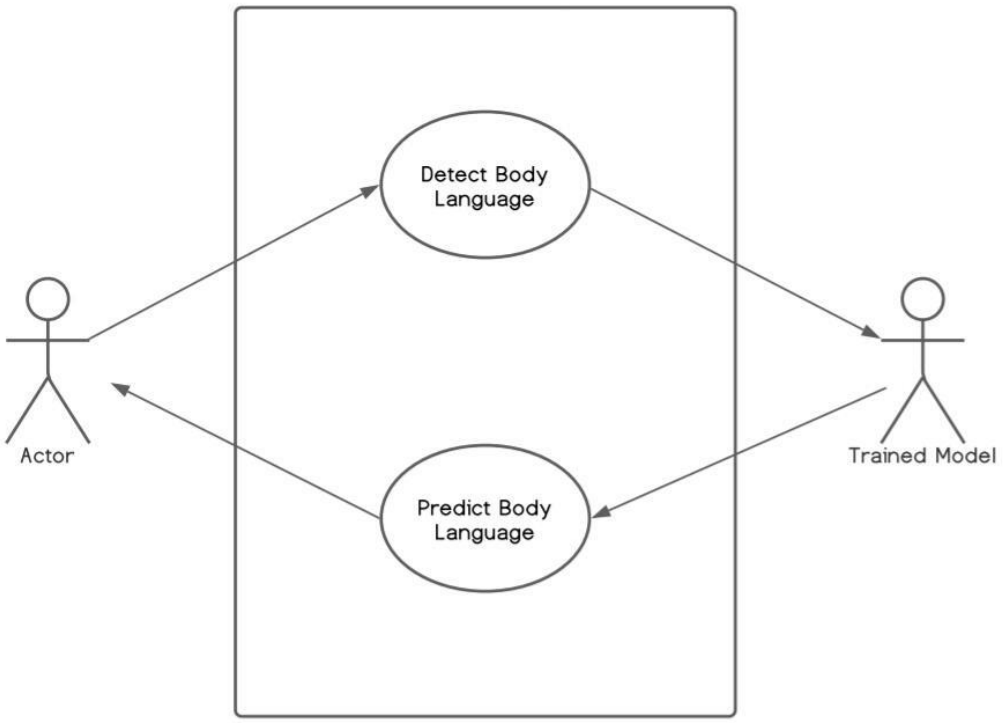
Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions that some systems should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.[11]

Graphical Notation:

The basic components of Use Case diagrams are the Actor, the Use Case, and the Association the Boundary boxes.

Table 4.1 Graphical Notations for Use Case Diagram

Actor	An Actor as mentioned is a user of the system and is depicted using a stick figure. The role of the user is written beneath the icon. Actors are not limited to humans. If a system communicates with another application and expects input or delivers output then that application can also be considered as an actor.	
Use case	A Use Case is the functionality provided by the system typically described as verb+ object (example: Register Car, Delete User). Use Cases are depicted with an ellipse. The name of the Use Case Is written within the ellipse.	
Directed Association	Associations are used to link Actors with use cases and indicate that an actor participates in the Use Case in some form. Directed Association is same as association but difference is that it represented by a line having an arrow head.	
System boundary boxes	You can draw a rectangle around the use cases, called the system boundary box, to indicate the scope of your system. Anything within the box represents functionality that is in scope and anything outside the box is not.	



4.4 System Requirements:

System requirements specification is a detailed statement of the effects that a system is required to achieve. A good specification gives a complete statement of what the system is to do, without making any commitment as to how the system is to do it. A system requirements specification is normally produced in response to a user requirements specifications or other expression of requirements, and is then used as the basis for system design. The system requirements specification typically differs from expression of requirements in both scope and precision the latter may cover both the envisaged system and the environment in which it will operate, but may leave many broad concepts unrefined.

4.4.1 Software Requirements:

- Operating System : Windows 7 and above.
- Language used : Python (3.9.0)
- Data Base : Excel sheet
- Libraries : Media pipe(version o.8.6), OpenCV(version 4.5.2), Pandas(1.3.0), Scikit-learn(0.24.2), Pickle5 (0.0.11)

4.4.2 Hardware Requirements:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Ram : 4 GB.
- webcam

CHAPTER – 5
SYSTEM IMPLEMENTATION

5. SYSTEM IMPLEMENTATION

5.1 TECHNOLOGY DESCRIPTION:

5.1.1 Windows[12]:

Windows is a series of operating systems developed by Microsoft. Each version of Windows includes a graphical user interface, with a desktop that allows users to view files and folders in windows. For the past two decades, Windows has been the most widely used operating system for personal computers PCs.

Features:

1. **Speed**

Even aside from incompatibilities and other issues that many people had with Vista, one of the most straightforward was speed – it just felt too sluggish compared to XP, even on pumped up hardware. Windows7brings a more responsive and sprightlier feel and Microsoft has spent a lot of time and effort getting the Start Menu response just right.

Microsoft has also recognized the need for improved desktop responsiveness, which gives the impression that the computer is responding to the user and that they are in control – something that was often lacking with Vista.

You can also expect faster boot times. And the boot sequence is now not only prettier than it was with Vista, but it's speedier too.

2. **Compatibility**

In simple terms, compatibility on Windows 7 will be far better than it was with Vista. Many programs that individuals and companies used on Windows XP did not work immediately and required updates, but with Windows 7 almost all applications that work on Vista should still run.

3. **Lower Hardware Requirements**

Vista gained a reputation for making even the beefiest hardware look rather ordinary. Windows 7, however, will run well on lower end hardware, making the transition from Window XP less painful.

Microsoft is even pushing Windows 7for netbooks. This could provide a modern replacement for Windows XP, which has found a new lease of life as the OS of choice on netbooks, supplanting Linux. The downside is that Windows 7 Starter Edition, as it will be called, will be limited to only three applications running at the same time.

4. Search and Organization

One of the best things about Windows 7 is the improved search tool, which now rivals

Mac OS X's Spotlight to be able to find what you need quickly and easily. For example, typing 'mouse' will bring up the mouse option within the control panel or typing a word will display it and split it up neatly into files, folders and applications.

5. Safety and Security

New security features in Windows include two new authentication methods tailored towards touchscreens (PINs and picture passwords), the addition of antivirus capabilities to Windows Defender (bringing it in parity with Microsoft Security Essentials) Smart Screen filtering integrated into Windows, and support for the "Secure Boot" functionality on UEFI systems to protect against malware infecting the boot process. Family Safety offers Parental controls, which allows parents to monitor and manage their children's activities on a device with activity reports and safety controls.

5.1.2 Anaconda (Python Distribution)[13]:

Anaconda is a free and open source distribution of the python and R programming Language for scientific computing such as Data Science, Machine Learning applications, large-scale data processing, predictive analytics etc.

Anaconda Navigator: Anaconda Navigator is a desktop Graphical User Interface includes in Anaconda Distribution that allows users to launch applications and manage

conda packages, environments and challenges without using command line commands . Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them.

The following Applications are available by default in navigator:

- ➢ Jupyter Lab
- ➢ Jupyter NoteBook
- ➢ QtConsole
- ➢ Spyder
- ➢ Glue
- ➢ Orange
- ➢ RStudio
- ➢ Visual Studio Code.

Benefits of Using Python Anaconda:

- v It is free and open-source
- v It has more than 1500 Python packages
- v Anaconda simplifies package management and deployment
- v It has tools to easily collect data from sources using machine learning and AI
- v It creates an environment that is easily manageable for deploying any project
- v Anaconda is the industry standard for developing, testing and training on a single machine
- v It has good community support- you can ask your questions there.

What you get:

- v Download more than 1500 Python/R data science packages
- v Manage libraries, dependencies, and environments with conda
- v Build and train ML and deep learning models with scikit-learn, TensorFlow and Theano

- Use Dask, NumPy, Pandas and Numba to analyze data scalably and fast
- Perform visualization with Matplotlib, Bokeh, Datashader, and Holoviews

Jupyter Notebook[14]:

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Getting Up and Running With Jupyter Notebook:

The Jupyter Notebook is not included with Python, so if you want to try it out, you will need to install Jupyter.

There are many distributions of the Python language. This will focus on just two of them for the purposes of installing Jupyter Notebook. The most popular is CPython, which is the reference version of Python that you can get from their website. It is also assumed that you are using Python 3.

5.1.3 Python[15]:

Python is a general-purpose interpreted, interactive, object-oriented, and high level Programming language. Python is designed to be highly readable. It uses English keywords frequently where as other languages. Python is easy to learn yet powerful and versatile scripting language, which makes it for Application development.

- **Python is interpreted:** Python is processed at runtime by the interpreter. You don't need to compile your programs before executing it. This is similar to PHP.

- **Python is Interactive:** you can actually sit at python prompt and interact with the interpreter directly to write your programs.
- **Python is Object Oriented**: Python supports Object-oriented style programming that encapsulates code within objects.
- **Python is Beginner's Language**: Python is a great language for the beginner-level and support development of a wide range applications from simple text processing to WWW browsers to games.

Applications of Python:

- **Web Applications:** It Provides Libraries to handle internet protocols such as HTML and XML, JSON, Email Processing, request, beautiful Soup, FeedParser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develop web based applications . Some important developments are: Python Wiki Engines, Pocoo, Python Blog Software
- **Desktop GUI Applications**: Python provides Tk GUI Library to develop user interface in python based application. Some other toolkits such as wxWidgets, Kivy, pyqt that are useable on several platforms. The kivy is popular for writing mulitouch applications.
- **Software Development**: Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.
- **BusinessApplications**: Python is used to build Business applications like ERP and e-commerce systems. Tryton is a high level application platform.
- **ConsoleBasedApplications**: We can use Python to develop console based applications.eg: IPYTHON

- **Audioor Video Based Applications**: Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of the real applications are: TimePlayer,cplay etc.,
- **3D CAD Applications**: To create CAD applications Fandango is a real application which provides full features of CAD.
- **Enterprise Application**: Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications:
  OpenErp, Tryton, Picalo.etc,.

Features of Python Programming[16]:

1. **Easy to code:**

Python is high level programming language. Python is very easy to learn language as compared to other language like c, c#, java script, java etc. It is very easy to code in python language and anybody can learn python basic in a few hours or days. It is also a developerfriendly language.

2. **Free and Open Source:**

Python language is freely available at the official website. Since, it is open-source, this means that source code is also available to the public. So you can download it as, use it as well as share it.

3. **Object-Oriented Language:**

One of the key features of python is Object-Oriented programming. Python supports object oriented language and concepts of classes, objects encapsulation etc.

4. **GUI Programming Support:**

Graphical Users interfaces can be made using a module such as PyQt5, PyQt4, wxPython or Tk in python.

PyQt5 is the most popular option for creating graphical apps with Python.

5. **High-Level Language:**

Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

6. **Extensible feature:**

Python is an Extensible language. We can write our python code into C or C++ language and also we can compile that code in C/C++ language.

7. **Python is Portable language:**

Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix and Mac then we do not need to change it, we can run this code on any platform.

8. **Python is Integrated language:**

Python is also an Integrated language because we can easily integrate Python with other languages like C, C++ etc.

9. **Interpreted Language:**

Python is an Interpreted Language because python code is executed line by line at a time. Unlike other languages C, C++, Java etc. there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called Bytecode.

10. Large Standard Library:

Python has a large standard library which provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers etc.

11. Dynamically Typed Language:

Python is dynamically-typed language. That means the type (for example- int, double, long etc) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

Libraries used:

- **os**- The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux[17].
- **mediapipe-** mediapipe is an open-source framework to "build word-class machine learning solutions" by Google — currently in the alpha stage. It has been open-sourced for a year now but has likely been under development for far longer[18].
- **cv2-** OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

  cv2.imread() method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix[19].
- **Pandas**- In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series[20].

- **Numpy-** NumPy is a Python package which stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful ndimensional array object, provide tools for integrating C, C++ etc.TheNumPy arrays takes significantly less amount of memory as compared to python lists. It also provides a mechanism of specifying the data types of the contents, which allows further optimisation of the code[21].
- **Pickle-** Pickle is used for serializing and de-serializing Python object structures, also called marshalling or flattening. Serialization refers to the process of converting an object in memory to a byte stream that can be stored on disk or sent over a network. Later on, this character stream can then be retrieved and de-serialized back to a Python object[22].
- **SK Learn-** Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib[23].

5.2 SYSTEM MODULES

1. **Install and import dependencies**

   In this step MediaPipe, OpenCV, pandas and scikit-learn are installed and imported.

2. **Make some detections**

   Here a webcam window is opened using cv2 and specifications like color and thickness of our face, hand and pose landmarks are mentioned/modified.

3. **Capture landmarks and export to CSV**

   Here various coordinates of facial expressions and body gestures are captured and exported to a csv file named coords.csv.

4. **Train custom model using scikit learn**

   Here the collected data is read and processed to train our machine learning classification model on it. The model is then evaluated and serialized.

5. **Make detections with model**

   Now the code, when run, can make predictions of the user's gestures using the above trained model.

CHAPTER – 6

TESTING

6.TESTING

6.1 Introduction:

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).[24]

Testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test have the following

- Meets the requirements that guided its design and development,
- Responds correctly to all kinds of inputs,
- Performs its functions within an acceptable time,
- Is sufficiently usable,
- Can be installed and run in its intended environments, and
- Achieves the general result its stakeholder's desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects).

Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors. It can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable.

Testing Strategies

Four Testing Strategies that are often adopted by the software development team include:

- Unit Testing
- Integration Testing

- Validation Testing
- System Testing

Unit Testing

We adopt white box testing when using this testing technique. This testing was carried out on individual components of the software that were designed. Each individual module was tested using this technique during the coding phase. Every component was checked to make sure that they adhere strictly to the specifications spelt out in the data flow diagram and ensure that they perform the purpose intended for them.

All the names of the variables are scrutinized to make sure that they are truly reflected of the element they represent. All the looping mechanisms were verified to ensure that they were as decided. Beside these, we trace through the code manually to capture syntax errors and logical errors.

Integration Testing

After finishing the Unit Testing process, next is the integration testing process. In this testing process we put our focus on identifying the interfaces between components and their functionality as dictated by the DFD diagram.

The Bottom up incremental approach was adopted during this testing. Low level modules are integrated and combined as a cluster before testing. The Black box testing technique was employed here. The interfaces between the components were tested first. This allowed identifying any wrong linkages or parameters passing early in the development process as it just can be passed in a set of data and checked if the result returned is an accepted one.

Validation Testing

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

System Testing

System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all the work should verify that all system elements have been properly integrated and perform allocated functions. System testing also ensures that the project works well in the environment. It traps the errors and allows convenient processing of errors without coming out of the program abruptly.

Recovery testing is done in such a way that failure is forced to a software system and checked whether the recovery is proper and accurate. The performance of the system is highly effective. Software testing is critical element of software quality assurance and represents ultimate review of specification, design and coding. Test case design focuses on a set of technique for the creation of test cases that meet overall testing objectives. Planning and testing of a programming system involve formulating a set of test cases, which are similar to the real data that the system is intended to manipulate. Test castes consist of input specifications, a description of the system functions exercised by the input and a statement of the extended output. In principle, testing of a program must be extensive. Every statement in the program should be exercised and every possible path combination through the program should be executed at least once. Thus, it is necessary to select a subset of the possible test cases and conjecture that this subset will adequately test the program.

Guidelines for developing test cases:

- Describe which feature or service your test attempts to cover.
- If the test case is based on a use case it is a good idea to refer to the use case name
- . Remember that the use cases are the source of test cases. In theory the software is supposed to match the use cases not the reverse. As soon as you have enough use cases , go ahead and write the test plan for that piece
- Specify what you are testing and which particular feature. Then specify whatϖ you are going to do to test the feature and what you expect to happen.
- Test the normal use of the object's methods. Test the abnormal but reasonableϖ use of the object's methods.
- Test the abnormal but unreasonable use of the object's methods.
- Test the boundary conditions. Also specify when you expect error dialogϖ boxes, when you expect some default event, and when functionality till is being defined.

- Test object's interactions and the messages sent among them. If you have developed sequence diagrams, they can assist you in this process.
- when the revisions have been made, document the cases so they become the starting bases for the follow- up test.

Attempting to reach agreement on answers generally will raise other what-if questions. Add these to the list and answer them, repeat the process until the list is stabilized, then you need not add any more questions. [25]

6.2 Test Cases:

Table 6.1 Test Cases Representation

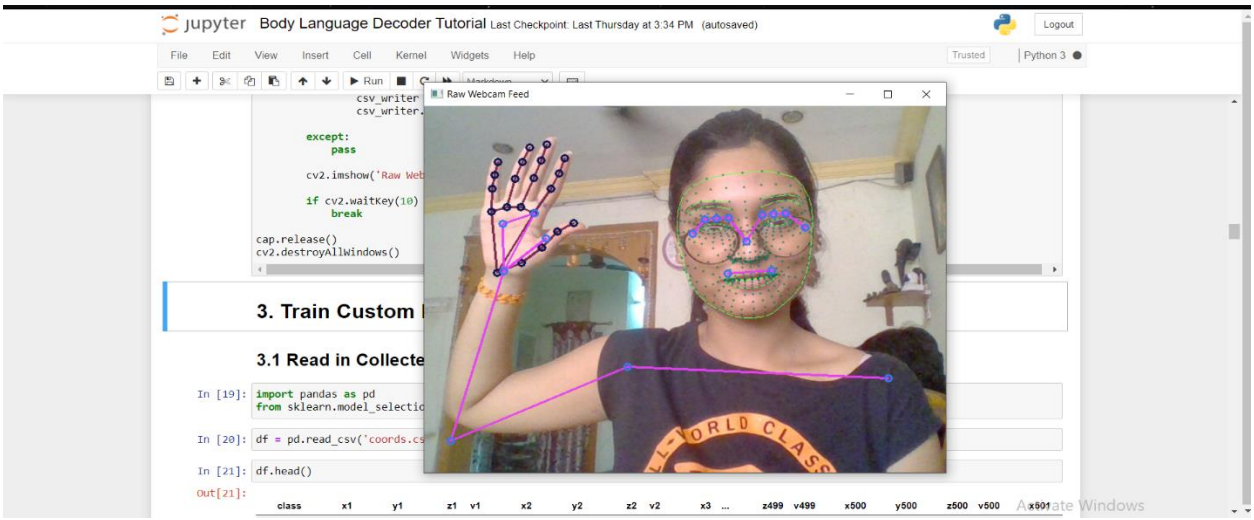
S.No	Description	Input	Expected Value	Actual Value	Result
1	Predicting a Happy face as happy	Happy face Through Web cam	Happy With probability	Happy Prob:0.97	PASS
2	Predicting a Sad face as sad	Sad face through web cam	Sad with probability	Sad Prob:0.99	PASS
3	Predicting a Victorious Gesture.	Victorious gestures through Web cam	Victory With probability	Victory Prob:0.92	PASS
4	Predicting Okay gesture.	Okay gesture Through web cam	Okay with probability	Okay Prob:0.97	PASS

CHAPTER – 7

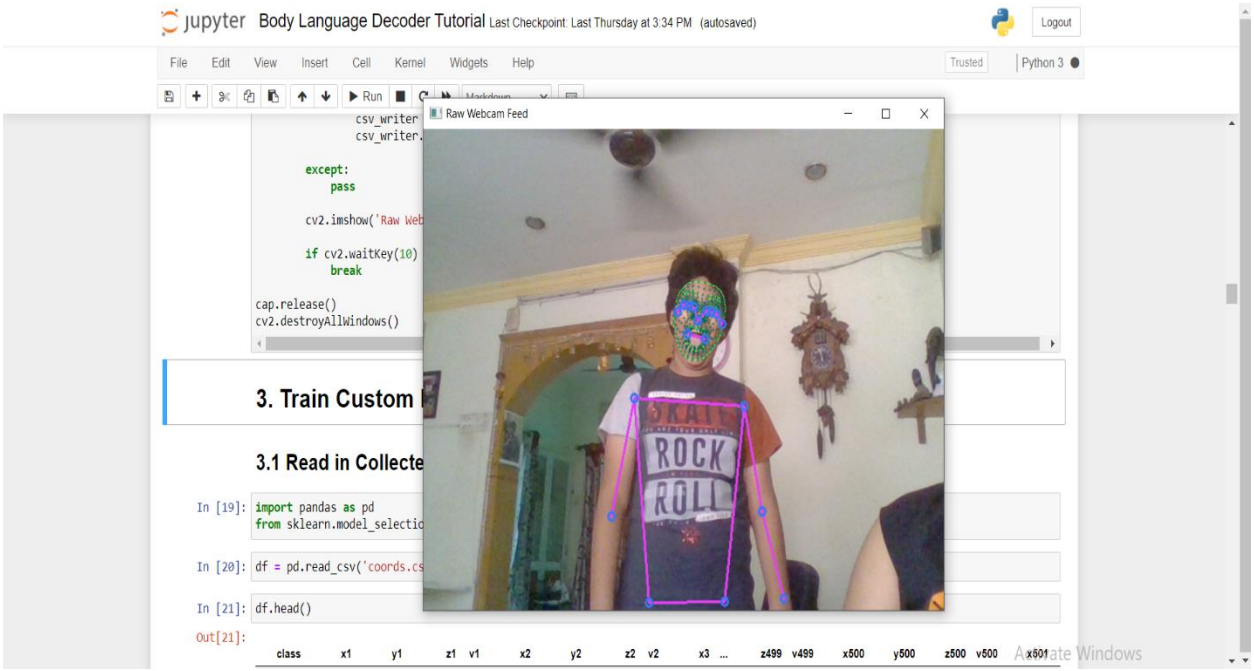
OUTPUT

7. OUTPUT SCREENS

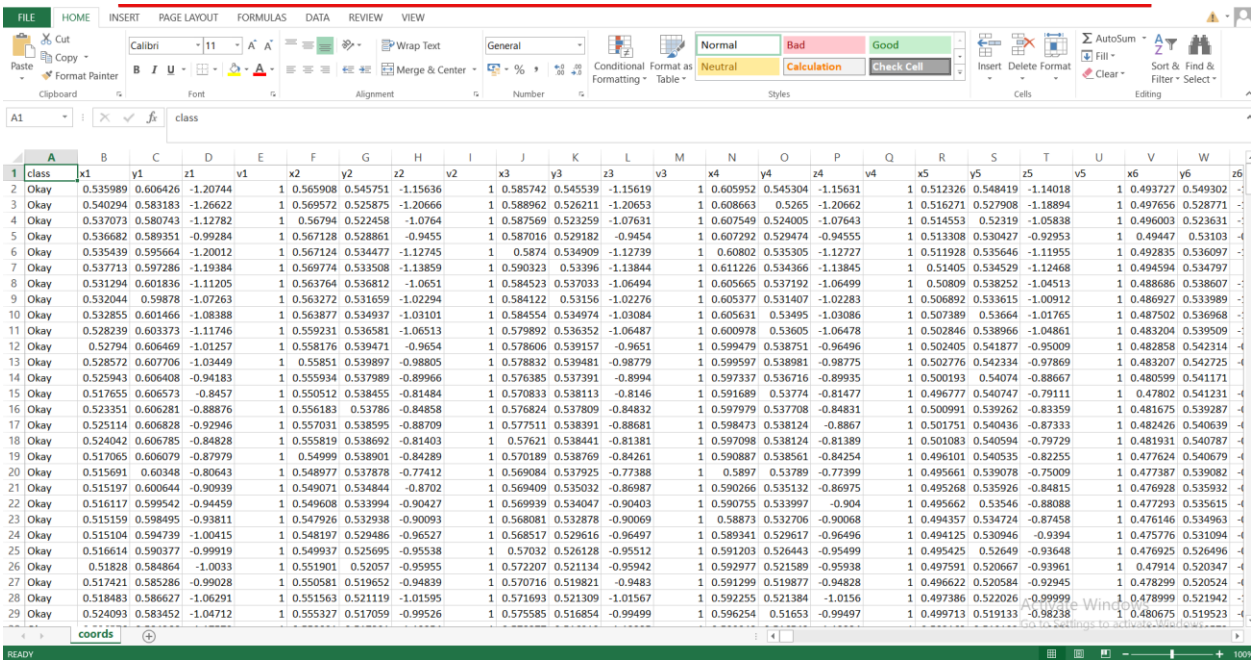
HAND LANDMARKS -



BODY LANDMARKS -

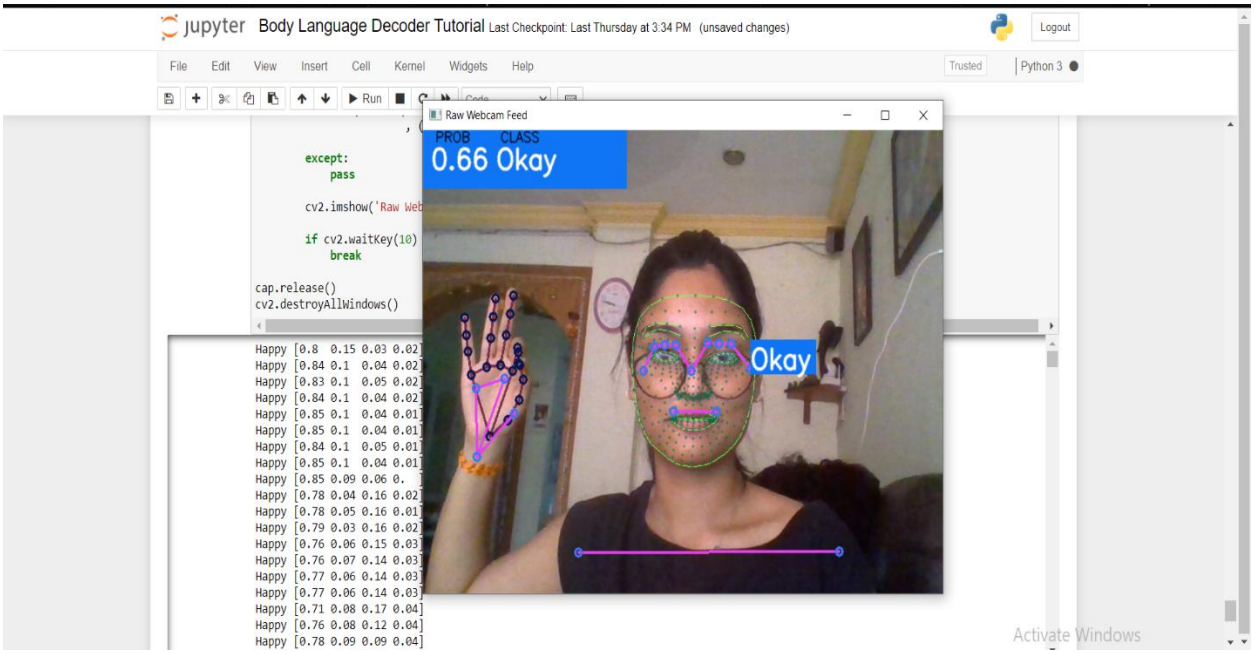


COORDS.CSV -

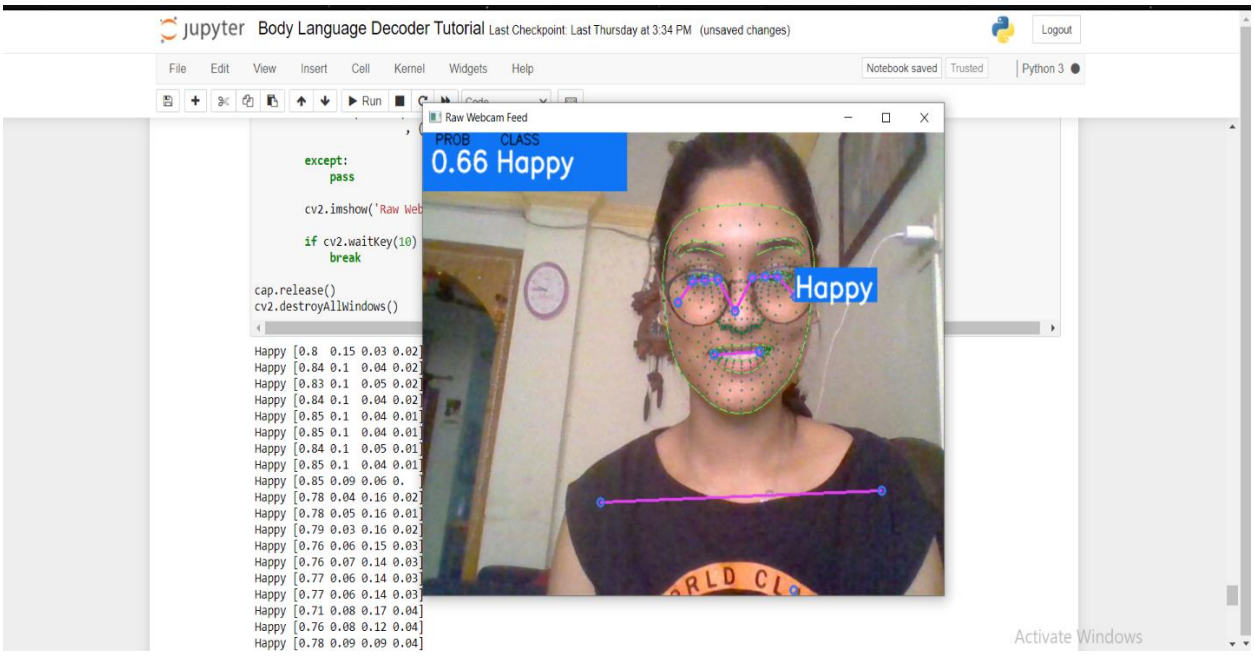


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	class	x1	y1	z1	v1	x2	y2	z2	v2	x3	y3	z3	v3	x4	y4	z4	v4	x5	y5	z5	v5	x6	y6	z6
2	Okay	0.535989	0.606426	-1.20744	1	0.565908	0.545751	-1.15636	1	0.585742	0.545539	-1.15619	1	0.605952	0.545304	-1.15631	1	0.512326	0.548419	-1.14018	1	0.493727	0.549302	-1
3	Okay	0.540294	0.583183	-1.26622	1	0.569572	0.525875	-1.20666	1	0.588962	0.526211	-1.20653	1	0.608663	0.5265	-1.20662	1	0.516271	0.527908	-1.18894	1	0.497656	0.528771	-1
4	Okay	0.537073	0.580743	-1.12782	1	0.56794	0.522458	-1.0764	1	0.587569	0.523259	-1.07631	1	0.607549	0.524005	-1.07643	1	0.514553	0.52319	-1.05838	1	0.496003	0.523631	-1
5	Okay	0.536682	0.589351	-0.99284	1	0.567128	0.528861	-0.9455	1	0.587016	0.529182	-0.9454	1	0.607292	0.529474	-0.94555	1	0.513308	0.530427	-0.92953	1	0.49447	0.53103	-1
6	Okay	0.535439	0.595664	-1.20012	1	0.567124	0.534477	-1.12745	1	0.5874	0.534909	-1.12739	1	0.60802	0.535305	-1.12727	1	0.511928	0.535646	-1.11955	1	0.492835	0.536097	-1
7	Okay	0.537713	0.597286	-1.19384	1	0.569774	0.533508	-1.13859	1	0.590323	0.53396	-1.13844	1	0.611226	0.534366	-1.13845	1	0.51405	0.534529	-1.12468	1	0.494594	0.534797	-1
8	Okay	0.531294	0.601836	-1.11205	1	0.563764	0.536812	-1.0651	1	0.584523	0.537033	-1.06494	1	0.605665	0.537192	-1.06499	1	0.50809	0.538252	-1.04513	1	0.488686	0.538607	-1
9	Okay	0.532044	0.59878	-1.07263	1	0.563272	0.531659	-1.02294	1	0.584122	0.53156	-1.02276	1	0.605377	0.531407	-1.02283	1	0.506892	0.533615	-1.00912	1	0.486927	0.533989	-1
10	Okay	0.532855	0.601466	-1.08388	1	0.563877	0.534937	-1.03101	1	0.584554	0.534974	-1.03084	1	0.605631	0.53495	-1.03086	1	0.507389	0.53664	-1.01765	1	0.487502	0.536968	-1
11	Okay	0.528239	0.603373	-1.11746	1	0.559231	0.536581	-1.06513	1	0.579892	0.536352	-1.06487	1	0.600978	0.53605	-1.06478	1	0.502846	0.538966	-1.04861	1	0.483204	0.539509	-1
12	Okay	0.52794	0.606469	-1.01257	1	0.558176	0.539471	-0.9654	1	0.578606	0.539157	-0.9651	1	0.599479	0.538751	-0.96496	1	0.502405	0.541877	-0.95009	1	0.482858	0.542314	-0
13	Okay	0.528572	0.607706	-1.03449	1	0.55851	0.539897	-0.98805	1	0.578832	0.539481	-0.98779	1	0.599597	0.538981	-0.98775	1	0.502776	0.542334	-0.97869	1	0.483207	0.542725	-0
14	Okay	0.525943	0.606408	-0.94183	1	0.555934	0.537989	-0.89966	1	0.576385	0.537391	-0.8994	1	0.597337	0.536716	-0.89935	1	0.500193	0.54074	-0.88667	1	0.480599	0.541171	-0
15	Okay	0.517655	0.606573	-0.8457	1	0.550512	0.538455	-0.81484	1	0.570833	0.538113	-0.8146	1	0.591689	0.53774	-0.81477	1	0.496777	0.540747	-0.79111	1	0.47802	0.541231	-0
16	Okay	0.523351	0.606281	-0.88876	1	0.556183	0.53786	-0.84858	1	0.576824	0.537809	-0.84832	1	0.597979	0.537708	-0.84831	1	0.500991	0.539262	-0.83359	1	0.481675	0.539287	-0
17	Okay	0.525114	0.606828	-0.92946	1	0.557031	0.538595	-0.88709	1	0.577511	0.538391	-0.88681	1	0.598473	0.538124	-0.8867	1	0.501751	0.540436	-0.87333	1	0.482426	0.540639	-0
18	Okay	0.524042	0.606785	-0.84828	1	0.555819	0.538692	-0.81403	1	0.57621	0.538441	-0.81381	1	0.597098	0.538124	-0.81389	1	0.501083	0.540594	-0.79729	1	0.481931	0.540787	-0
19	Okay	0.517065	0.606079	-0.87979	1	0.54999	0.538901	-0.84289	1	0.570189	0.538769	-0.84261	1	0.590887	0.538561	-0.84254	1	0.496101	0.540535	-0.82255	1	0.477624	0.540679	-0
20	Okay	0.515691	0.60348	-0.80643	1	0.548977	0.537878	-0.77412	1	0.569084	0.537925	-0.77388	1	0.5897	0.53789	-0.77399	1	0.495661	0.539078	-0.75009	1	0.477387	0.539082	-0
21	Okay	0.515197	0.600644	-0.90939	1	0.549071	0.534844	-0.8702	1	0.569409	0.535032	-0.86987	1	0.590266	0.535132	-0.86975	1	0.495268	0.535926	-0.84815	1	0.476928	0.535932	-0
22	Okay	0.516117	0.599542	-0.94459	1	0.549608	0.533994	-0.90427	1	0.569939	0.534047	-0.90403	1	0.590755	0.533997	-0.904	1	0.495662	0.53546	-0.88088	1	0.477293	0.535615	-0
23	Okay	0.515159	0.598495	-0.93811	1	0.547926	0.532938	-0.90093	1	0.568081	0.532878	-0.90069	1	0.58873	0.532706	-0.90068	1	0.494357	0.534724	-0.87458	1	0.476146	0.534963	-0
24	Okay	0.515104	0.594739	-1.00415	1	0.548197	0.529486	-0.96527	1	0.568517	0.529616	-0.96497	1	0.589341	0.529617	-0.96496	1	0.494125	0.530946	-0.9394	1	0.475776	0.531094	-0
25	Okay	0.516614	0.590377	-0.99919	1	0.549937	0.525695	-0.95538	1	0.57032	0.526128	-0.95512	1	0.591203	0.526443	-0.95499	1	0.495425	0.52649	-0.93648	1	0.476925	0.526496	-0
26	Okay	0.51828	0.584864	-1.0033	1	0.551901	0.52057	-0.95955	1	0.572207	0.521134	-0.95942	1	0.592977	0.521589	-0.95938	1	0.497591	0.520667	-0.93961	1	0.47914	0.520347	-0
27	Okay	0.517421	0.585286	-0.99028	1	0.550581	0.519652	-0.94839	1	0.570716	0.519821	-0.9483	1	0.591299	0.519877	-0.94828	1	0.496622	0.520584	-0.92945	1	0.478299	0.520524	-0
28	Okay	0.518483	0.586627	-1.06291	1	0.551563	0.521119	-1.01595	1	0.571693	0.521309	-1.01567	1	0.592255	0.521384	-1.0156	1	0.497386	0.522026	-0.99999	1	0.478999	0.521942	-1
29	Okay	0.524093	0.583452	-1.04712	1	0.555327	0.517059	-0.99526	1	0.575585	0.516854	-0.99499	1	0.596254	0.51653	-0.99497	1	0.499713	0.519133	-0.98238	1	0.480675	0.519523	-0

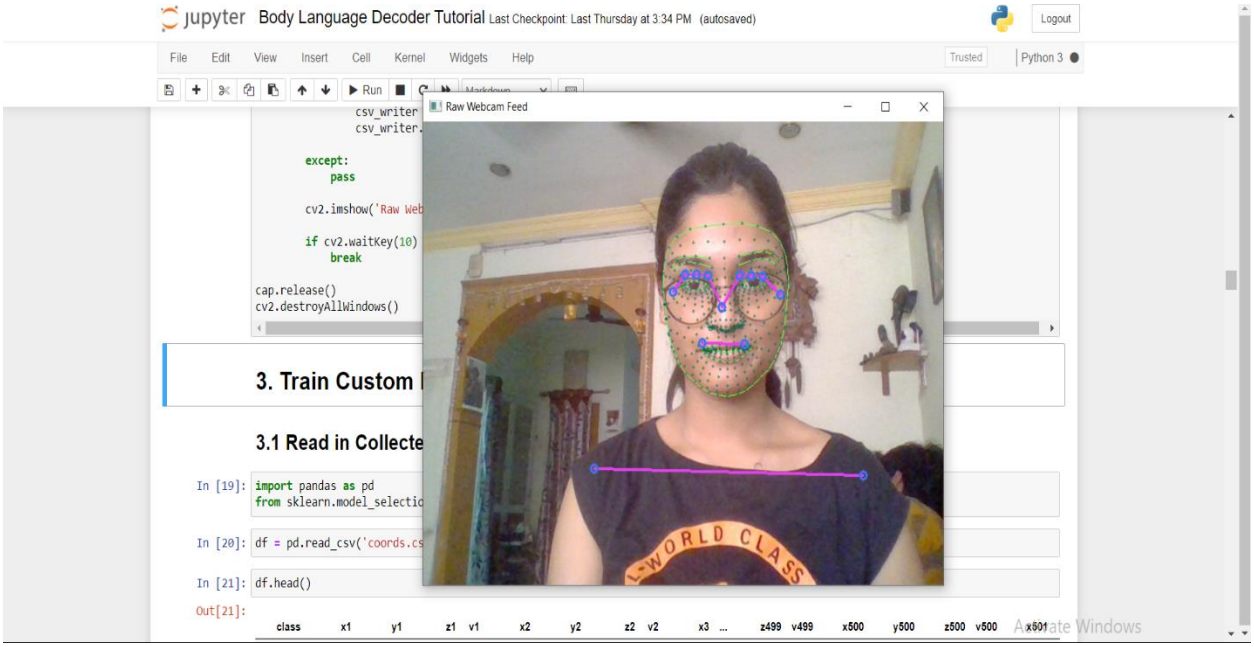
“OKAY” EXPRESSION –



“HAPPY ” EXPRESSION –



FACE LANDMARKS -



CHAPTER – 8

CONCLUSION AND FUTURE SCOPE OF THE WORK

8. CONCLUSION AND FUTURE SCOPE OF THE WORK

Detecting and analyzing body language is gaining a lot of attention lately. Being able to detect and analyze facial expressions of client/customer helps businesses and marketing teams to get honest reviews and feedbacks. But facial expression is just a small part of body language. Body language consists of others elements like hand gestures and body poses. And body language plays a very important role in communication. For example in interviews, interviewers take candidate's body language into consideration. By enhancing this project, a tool can be provided to the interviewers which aids them in understanding how the candidate is responding when asked questions from different domains or put in different situations during HR rounds. Since this project supports real time hand landmark detection, hand sign language detection can also be implemented. Not only that, using this project, implementation of already existing projects like drowsiness detection of drivers, action detection etc can be made easier with much better results.

REFERENCES –

[1] https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html

[2] https://heartbeat.fritz.ai/simultaneously-detecting-face-hand-motion-and-pose-in-real-time-on-mobile-devices-27849560fc4e

[3] https://medium.com/jstack-eu/using-machine-learning-to-analyse-body-language-and-facial-expressions-a779172cc98

[4] Danilo Avola, Luigi Cinque, Stefano Levialdi, Giuseppe Placidi, International Conference on Image Analysis and Processing, ICIAP 2013: New Trends in Image Analysis and Processing – ICIAP 2013 pp 465-473

[5] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg and Matthias Grundmann, arXiv:1906.08172v1 [cs.DC] 14 Jun 2019

[6] Fabina pedregosa, Gael varoquaux, alexandre Gramfort, Vincent Michel, Bertrand Thirion, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research 12 (2011) 2825-2830

[7] https://creately.com/blog/diagrams/uml-diagram-types-examples/

[8] https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/

[9] https://en.m.wikipedia.org/wiki/Non-functional_requirement

[10] https://mymanagementguide.com/feasibility-study-template/amp/

[11] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

[12] https://www.itpro.co.uk/610706/the-top-10-microsoft-windows-7-features

[13] An introduction to Anaconda: what it is, and how to install it (freecodecamp.org)

[14] Jupyter Notebook: An Introduction – Real Python

[15] Python Language Introduction - GeeksforGeeks

[16] Python Features - GeeksforGeeks

[17] OS Module in Python with Examples - GeeksforGeeks

[18] MediaPipe: A Framework for Building Perception Pipelines | DeepAI

[19] Introduction to OpenCV - GeeksforGeeks

[20] Python Pandas - Introduction - Tutorialspoint

[21] NumPy Installation - Introduction to NumPy - Numeric Python (machinelearning.org.in)

[22] Serializing Python Objects Using Python's Pickle Module (techbeamers.com)

[23] Scikit Learn - Introduction - Tutorialspoint

[24] https://www.guru99.com/software-testing-introduction-importance.html

[25] https://blog.testlodge.com/how-to-write-test-cases-for-software-with-sample/

APPENDIX –

```
!pip install mediapipe opencv-python pandas scikit-learn
import mediapipe as mp # Import mediapipe
import cv2 # Import opencv
import csv
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score # Accuracy metrics
import pickle
mp_drawing = mp.solutions.drawing_utils # Drawing helpers
mp_holistic = mp.solutions.holistic # Mediapipe Solutions
cap = cv2.VideoCapture(0)
# Initiate holistic model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5)
as holistic:
    while cap.isOpened():
    ret, frame = cap.read()
    # Recolor Feed
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    image.flags.writeable = False
    # Make Detections
    results = holistic.process(image)
    # print(results.face_landmarks)
    # face_landmarks, pose_landmarks, left_hand_landmarks, right_hand_landmarks
    # Recolor image back to BGR for rendering
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    # 1. Draw face landmarks
    mp_drawing.draw_landmarks(image, results.face_landmarks,
mp_holistic.FACE_CONNECTIONS,
                        mp_drawing.DrawingSpec(color=(80,110,10), thickness=1,
circle_radius=1),
                        mp_drawing.DrawingSpec(color=(80,256,121), thickness=1,
circle_radius=1)
                        )
    # 2. Right hand
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                        mp_drawing.DrawingSpec(color=(80,22,10), thickness=2,
circle_radius=4),
```

```
                        mp_drawing.DrawingSpec(color=(80,44,121), thickness=2,
circle_radius=2)
                        )
      # 3. Left Hand
       mp_drawing.draw_landmarks(image, results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                        mp_drawing.DrawingSpec(color=(121,22,76), thickness=2,
circle_radius=4),
                        mp_drawing.DrawingSpec(color=(121,44,250), thickness=2,
circle_radius=2)
                        )
      # 4. Pose Detections
      mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_holistic.POSE_CONNECTIONS,
                        mp_drawing.DrawingSpec(color=(245,117,66), thickness=2,
circle_radius=4),
                        mp_drawing.DrawingSpec(color=(245,66,230), thickness=2,
circle_radius=2)
                        )
      cv2.imshow('Raw Webcam Feed', image)
      if cv2.waitKey(10) & 0xFF == ord('q'):
         break
cap.release()
cv2.destroyAllWindows()
results.face_landmarks.landmark[0].visibility
num_coords =
len(results.pose_landmarks.landmark)+len(results.face_landmarks.landmark)
num_coords
landmarks = ['class']
for val in range(1, num_coords+1):
   landmarks += ['x{}'.format(val), 'y{}'.format(val), 'z{}'.format(val), 'v{}'.format(val)]
with open('coords.csv', mode='w', newline='') as f:
   csv_writer = csv.writer(f, delimiter=',', quotechar='"',
quoting=csv.QUOTE_MINIMAL)
   csv_writer.writerow(landmarks)
class_name = "Okay"
cap = cv2.VideoCapture(0)
# Initiate holistic model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5)
as holistic:
   while cap.isOpened():
      ret, frame = cap.read()
      # Recolor Feed
      image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
      image.flags.writeable = False
      # Make Detections
      results = holistic.process(image)
```

```
        # print(results.face_landmarks)
        # face_landmarks, pose_landmarks, left_hand_landmarks, right_hand_landmarks
        # Recolor image back to BGR for rendering
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        # 1. Draw face landmarks
        mp_drawing.draw_landmarks(image, results.face_landmarks,
mp_holistic.FACE_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(80,110,10), thickness=1,
circle_radius=1),
                                  mp_drawing.DrawingSpec(color=(80,256,121), thickness=1,
circle_radius=1)
                                  )
        # 2. Right hand
        mp_drawing.draw_landmarks(image, results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(80,22,10), thickness=2,
circle_radius=4),
                                  mp_drawing.DrawingSpec(color=(80,44,121), thickness=2,
circle_radius=2)
                                  )
       # 3. Left Hand
        mp_drawing.draw_landmarks(image, results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(121,22,76), thickness=2,
circle_radius=4),
                                  mp_drawing.DrawingSpec(color=(121,44,250), thickness=2,
circle_radius=2)
                                  )
        # 4. Pose Detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_holistic.POSE_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(245,117,66), thickness=2,
circle_radius=4),
                                  mp_drawing.DrawingSpec(color=(245,66,230), thickness=2,
circle_radius=2)
                                  )
        cv2.imshow('Raw Webcam Feed', image)
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
cap.release()
cv2.destroyAllWindows()
df = pd.read_csv('coords.csv')
X = df.drop('class', axis=1) # features
y = df['class'] # target value
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1234)
pipelines = {
```

```
    'lr':make_pipeline(StandardScaler(), LogisticRegression()),
    'rc':make_pipeline(StandardScaler(), RidgeClassifier()),
    'rf':make_pipeline(StandardScaler(), RandomForestClassifier()),
    'gb':make_pipeline(StandardScaler(), GradientBoostingClassifier()),
}
fit_models = {}
for algo, pipeline in pipelines.items():
    model = pipeline.fit(X_train, y_train)
    fit_models[algo] = model
fit_models['rc'].predict(X_test)
for algo, model in fit_models.items():
    yhat = model.predict(X_test)
    print(algo, accuracy_score(y_test, yhat))
fit_models['rf'].predict(X_test)
with open('body_language.pkl', 'wb') as f:
    pickle.dump(fit_models['rf'], f)
with open('body_language.pkl', 'rb') as f:
    model = pickle.load(f)
cap = cv2.VideoCapture(0)
# Initiate holistic model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5)
as holistic:
    while cap.isOpened():
        ret, frame = cap.read()
        # Recolor Feed
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False
        # Make Detections
        results = holistic.process(image)
        # print(results.face_landmarks)
        # face_landmarks, pose_landmarks, left_hand_landmarks, right_hand_landmarks
        # Recolor image back to BGR for rendering
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        # 1. Draw face landmarks
        mp_drawing.draw_landmarks(image, results.face_landmarks,
mp_holistic.FACE_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(80,110,10), thickness=1,
circle_radius=1),
                                  mp_drawing.DrawingSpec(color=(80,256,121), thickness=1,
circle_radius=1)
                                  )
        # 2. Right hand
        mp_drawing.draw_landmarks(image, results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(80,22,10), thickness=2,
circle_radius=4),
```

```
                          mp_drawing.DrawingSpec(color=(80,44,121), thickness=2,
circle_radius=2)
                          )
        # 3. Left Hand
        mp_drawing.draw_landmarks(image, results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                          mp_drawing.DrawingSpec(color=(121,22,76), thickness=2,
circle_radius=4),
                          mp_drawing.DrawingSpec(color=(121,44,250), thickness=2,
circle_radius=2)
                          )
        # 4. Pose Detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_holistic.POSE_CONNECTIONS,
                          mp_drawing.DrawingSpec(color=(245,117,66), thickness=2,
circle_radius=4),
                          mp_drawing.DrawingSpec(color=(245,66,230), thickness=2,
circle_radius=2)
                          )
        cv2.imshow('Raw Webcam Feed', image)
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
cap.release()
cv2.destroyAllWindows()
```

IJARIIT CERTIFICATES


INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY
CERTIFICATE OF PUBLICATION
This is to certify that
J. Sri Gayathri Seelamanthula
published a paper entitled
AI Body Language Decoder using MediaPipe and Python
Volume-7, Issue-3 - May-June, 2021
ISSN:2454-132X
IMPACT FACTOR 6.018
Ref No.OAP/IJ/213/1485
editor@ijariit.com, ijariit@gmail.com
Editor In Chief
Google Scholar
Cosmos
DRJI
RESEARCHERID
THOMSON REUTERS
IndianScience.in


INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY
CERTIFICATE OF PUBLICATION
This is to certify that
Madhurima kalivarapu
published a paper entitled
AI Body Language Decoder using MediaPipe and Python
Volume-7, Issue-3 - May-June, 2021
ISSN:2454-132X
IMPACT FACTOR 6.018
IJARIIT
Ref No.OAP/I1/213/1486
editor@ijariit.com, ijariit@gmail.com
Editor-In-Chief
Google Scholar
Cosmos
DRJI
RESEARCHERID
THOMSON REUTERS
IndianScience.in


INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY
CERTIFICATE OF PUBLICATION
This is to certify that
Sai Prathyusha Kanisetti
published a paper entitled
AI Body Language Decoder using MediaPipe and Python
Volume-7, Issue-3 - May-June, 2021
ISSN:2454-132X
IMPACT FACTOR 6.018
Ref No.OAP/IJ/213/1483
editor@ijariit.com, ijariit@gmail.com
Editor In Chief
Google Scholar
Cosmos
DRJI
RESEARCHERID
THOMSON REUTERS
IndianScience.in

INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

CERTIFICATE OF PUBLICATION

This is to certify that

Sankeerthana Rajan Karem

published a paper entitled

AI Body Language Decoder using MediaPipe and Python

Volume-7, Issue-3 - May-June, 2021

ISSN:2454-132X
IMPACT FACTOR 6.018

Ref No.OAP/I1/213/1482

editor@ijariit.com, ijariit@gmail.com

Editor In Chief

INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 7, Issue 3 - V7I3-2223)
Available online at: https://www.ijariit.com

AI Body Language Decoder using MediaPipe and Python

Sankeerthana Rajan Karem
sankeerthanakarem@gmail.com
Andhra University College of Engineering for Women, Visakhapatnam, Andhra Pradesh

Sai Prathyusha Kanisetti
saiprathyusha9121@gmail.com
Andhra University College of Engineering for Women, Visakhapatnam, Andhra Pradesh

Dr. K. Soumya
soumyacf@andhrauniversity.edu.in
Andhra University College of Engineering for Women, Visakhapatnam, Andhra Pradesh

J. Sri Gayathri Seelamanthula
sjsgayathri99@gmail.com
Andhra University College of Engineering for Women, Visakhapatnam, Andhra Pradesh

Madhurima Kalivarapu
madhurimak.123@gmail.com
Andhra University College of Engineering for Women, Visakhapatnam, Andhra Pradesh

ABSTRACT

Body language are visual languages produced by the movement of the hands, face and body. In this project we evaluate representations based on skeleton poses, as these are explainable, person-independent, privacy-preserving, low-Dimensional representations. Basically, skeletal representations generalize over an individual's appearance and background, allowing us to focus on the recognition of motion. We present a real-time on-device body tracking pipeline that predicts hand skeleton and the whole-body notion. It is implemented via MediaPipe, a framework for building cross-platform ML solutions. We perform using pose estimation systems and analyze the applicability of the estimation systems to body language recognition by evaluating failure cases of the existing models. The proposed system and architecture demonstrate real-time inference and high prediction quality.

Keywords— Body Landmarks, MediaPipe, Body Language, Prediction, Accuracy, Real-time on-device Tracking, Pose Estimation, Recognition.

1. INTRODUCTION

Body Language Decoder helps detect and predict facial expressions, hand gestures and body pose. Facial expression recognition can help market research companies scale data and analyze quickly. The ability to perceive the shape and motion of hands can be a vital component in improving the user experience across a variety of technological domains and platforms truly. For example, it can form the basis for sign language understanding and hand posture control, and can also enable the overlay of digital content and information on top of the physical world in augmented reality.

The MediaPipe Body Landmark model gives for high-fidelity body pose tracking, inferring 33 2D landmarks on the body (or 25 upper-body landmarks) from RGB video frames, utilizing BlazePose. It detects the landmarks of a every single body pose, full-body by default, but it can be configured to cover the upper-body only, in such case it only predicts the first 25 landmarks.

MediaPipe Hands is a high-fidelity hand and finger tracking solution. In Just a single frame in can infer up to 21 3D hand Landmarks. It's a hybrid between a palm/hand detection model that operates on the full image and returns an oriented hand bounding box and a hand landmark model that operates on the image that is cropped region which is defined by the palm detector, which returns high-fidelity 3D hand key points. It detects landmarks of a single hand or both hands depending on the module type.

MediaPipe Face Mesh estimates 468 3D face landmarks accurately in real-time on-tracking mobile devices. It employs deep neural networks to infer the 3D surface geometry, requiring only a single camera input, without the need for a dedicated depth sensor.

Applications: Driver drowsiness detection, Sign language detection, Market research companies can use this technology to analyze data, Body language detection in interviews.

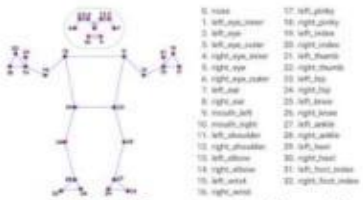


Fig 1.1 pose landmark

Fig 1.2 hand landmark

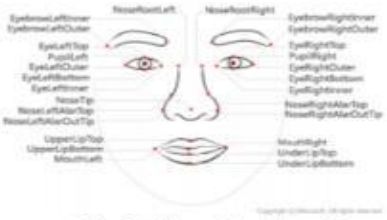


Fig 1.3 face landmark

Sentiment analysis is the process of detecting positive or negative sentiment in a sentence. It's often used by businesses to detect sentiment in social data, gauge brand reputation, and understand customers. Sentiment Analysis is already widely used by different companies to use towards their product or brand in the digital world. However, in the offline world users are also interacting with the brands and products in retail stores, showrooms, social media etc. and solutions to measure user's reaction/expression automatically under such settings has remained a challenging task. Emotion Detection from facial expressions or a text using AI can be a viable alternative to automatically measure consumer's engagement with their content and brands. On the other hand detecting body language, which is considered as one of the most effective communication method, can help interviewers analyze the candidate during the process.

This can be achieved by creating an excel sheet of coordinate points of landmarks of desired poses and training the model on that. This trained model can then be stored and used later for predicting user's gestures.

2. BACKGROUND AND RELATED WORK

In this section, we present the main components of what we call a Body Gesture Recognition system. An important preparation step, which influences all the subsequent design decisions for such an automatic pipeline is the determination of the appropriate modelling of input (human body/gesture) and targets (emotion/expressions). Live perception of simultaneous human gesture, face landmarks, and hand tracking in real-time on mobile devices can enable various modern life applications: fitness and sport analysis, posture control and sign language recognition, augmented reality try-on and effects. MediaPipe already offers immediate, fast and accurate, yet separate, solutions for these complex tasks. Combining them all into a real-time semantically consistent end-to-end solution is a uniquely difficult problem as requiring simultaneous inference of multiple, dependent neural networks.

The MediaPipe Holistic pipeline integrates separate models for body i.e structure , face and hand components, each of which are optimized for their particular domain. The pose estimation model, for example, takes a lower resolutions and fixed resolution video frame (256x256) as input resolutions. But if one were to crop the hand and face regions/parts from that image to pass to their respective models, the image resolution would be too low for accurate articulation. Therefore, we designed MediaPipe Holistic as a multi-stage pipeline, which is more accurate than any other, which treats the different regions using a region appropriate image resolution.

First, we estimate the human pose with BlazePose's pose detector and subsequent landmark model. Then, using the inferred pose landmarks we derive three regions of interest (ROI) crops for each hand gesture (2x) and the face expression, and employ a re-crop model to improve the ROI. We then crop the full-resolution input frame/coordinates to these ROIs and apply task-specific face and hand models to estimate their corresponding landmarks. Finally, we merge all landmarks with those of the pose model to yield the full body landmarks.

The pipeline is implemented as a MediaPipe graph that uses a holistic MediaPipe landmark subgraph from the holistic landmark module and renders using a dedicated holistic renderer subgraph. The holistic landmark subgraph internally uses a pose/body landmark module, hand landmark module and face landmark module.

To summarize, the related work mentioned above, this system gives the recognition to the complete body/shape/ surface. Using holistic MediaPipe the face, hand and as well as body posture is detected. It collects the coordinates, processing image using opencv machine learning library and merges them giving the required output using scikit-learn machine learning library.

3. PROPOSED SYSTEM

The flowchart of the proposed system is given below.

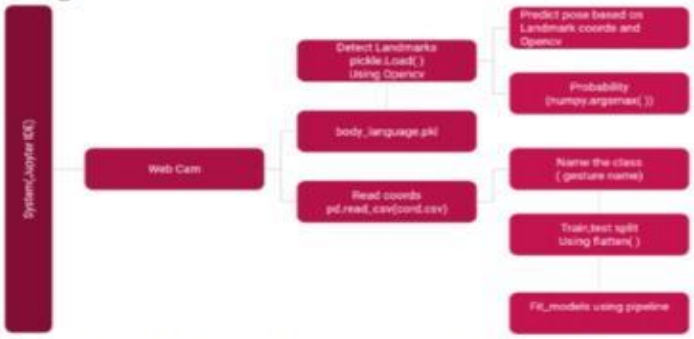


Fig 3.1: Flowchart of working model using IDE

3.1 Working

Initially we run code on jupyter IDE. when code is executed, camera starts running. It reads the coordinates using holistic MEDIAPIPE and then compares it with predefined body_language.pkl file when we trained earlier. Then it predicts the gesture. Also, it predicts accuracy of the pose comparing to pre-trained gesture using numpy-argsmax( ) function imported from Numpy library.

In case of training the model, we run the code and web cam starts running. It reads the co-ordinates from our body using holistic function imported from MediaPipe library and writes to coords.csv file. We name the class as the name of the gesture we are training. These coordinates are then clustered using flatten( ) function imported from scikit- learning ML library.

3.2 System Modules

(a) Install and import dependencies: In this step mediapipe, opencv, pandas and scikit-learn are installed and imported.

(b) Make some detections: Here a webcam window is opened using cv2 and specifications like color and thickness of our face, hand and pose landmarks are mentioned/modified.

(c) Capture landmarks and export to CSV: Here various coordinates of facial expressions and body gestures are captured and exported to a csv file named coords.csv.

(d) Train custom model using scikit learn: Here the collected data is read and processed to train our machine learning classification model on it.The model is then evaluated and serialized.

(e) Make detections with model: Now the code, when run, can make predictions of the user's gestures using the above trained model.

4. TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects). Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors.

Table 4.1 Test Cases Representation

S.No	Description	Input	Expected Value	Actual Value	Result
1	Predicting a Happy face as happy	Happy face Through Web cam	Happy With probability	Happy Prob:0.97	PASS
2	Predicting a Sad face as sad	Sad face through web cam	Sad with probability	Sad Prob:0.99	PASS
3	Predicting a Victorious Gesture.	Victorious gestures through Web cam	Victory With probability	Victory Prob:0.92	PASS
4	Predicting Okay gesture.	Okay gesture Through web cam	Okay with probability	Okay Prob:0.97	PASS

Software testing can be conducted as soon as executable software/program (even if partially complete) exists. The overall approach to software testing or development often determines when and how testing is conducted and results. For example, in a phased process, most testing process occurs after system requirements have been defined and then implemented in testable.

5. RESULTS

Fig 5.1 Hand landmarks

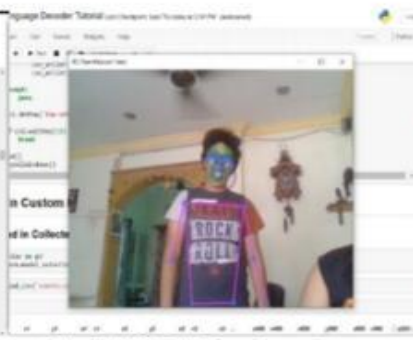


Fig 5.2 Body Landmarks

Fig 5.3 Face Landmarks

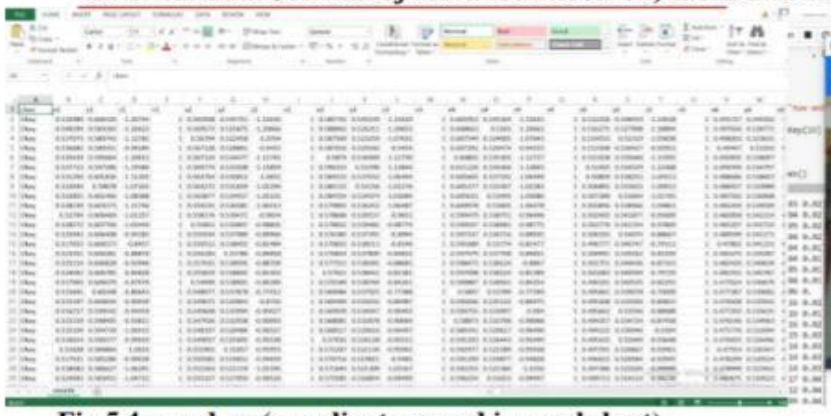


Fig 5.4 coord.csv(coordinates saved in excel sheet)

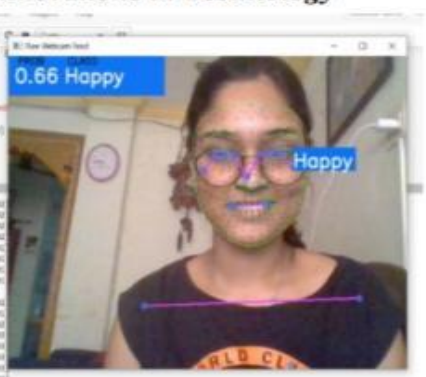


Fig 5.5 "happy" expression

6. CONCLUSION AND FUTURE SCOPE

Detecting and analyzing body language is gaining a lot of attention lately. Being able to detect and analyze facial expressions of client/customer helps businesses and marketing teams to get honest reviews and feedbacks. But facial expression is just a small part of body language. Body language consists of others elements like hand gestures and body poses. And body language plays a very important role in communication. For example in interviews, interviewers take candidate's body language into consideration. By enhancing this project, a tool can be provided to the interviewers which aids them in understanding how the candidate is responding when asked questions from different domains or put in different situations during HR rounds. Since this project supports real time hand landmark detection, hand sign language detection can also be implemented. Not only that, using this project, implementation of already existing projects like drowsiness detection of drivers, action detection etc can be made easier with much better results.

7. REFERENCES

[1] https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html
[2] https://heartbeat.fritz.ai/simultaneously-detecting-face-hand-motion-and-pose-in-real-time-on-mobile-devices-27849560fc4e
[3] https://medium.com/jstack-eu/using-machine-learning-to-analyse-body-language-and-facial-expressions-a779172cc98
[4] http://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html
[5] https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/
[6] https://www.guru99.com/software-testing-introduction-importance.html
[7] https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?source=post_page----------
[8] https://link.springer.com/chapter/10.1007/978-3-642-41190-8_50
[9] https://medium.com/jstack-eu/using-machine-learning-to-analyse-body-language-and-facial-expressions-a779172cc98
[10] https://arxiv.org/pdf/1906.08172.pdf

