AI-Driven Sport Analysis and Improvement System Final Report

Ivan Phua You Wen, *2202433*; Koh Yi Tong, *2201442*; Lai Chung Shing, *2203541*; Jeremy Lim Min En, *2203516*; Nicholas Phoon Kai Jin, *2200780*;

Name of the Industry Supervisor: **Choo Yun Joo**; name of the Company: **Starview Technologies**
Name of the SIT Academic Supervisor(s): **Purnima Murali Mohan** and **Peter Loh Kok Keong**

ABSTRACT

AI-Driven Sport Analysis and Improvement will be a system that revolutionises the way any golfer improves their performance with the power of artificial intelligence and machine learning. It uses cutting edge computer vision and motion capture technologies to evaluate each golfer's swing and body movement in real-time, with great feedback on performance and areas in need of improvement.

The system's development—from conception and design to implementation and testing—is presented in this final report. It also displays the difficulties faced, the approaches taken, and the outcomes attained. Case studies will shed more light on how the technology improves golfing abilities.

These modules would consist of a high-resolution camera and sensor setup to track key points on the body, AI algorithms running in a data processing backend, and a user-friendly interface. These will combine during exercise sessions to allow highly accurate biomechanical analysis—outlining areas for improvement and giving real-time feedback. One is also able to track long-term performance in order to establish progress and make suitable adjustments in training strategies.

This final report presents the development of the system from its conception and design to its implementation and testing. It also shows challenges encountered, methodologies used, and results achieved. User feedback and case studies will further enlighten how the system works effectively in enhancing golfing skills.

ACKNOWLEDGMENTS

Our team would like to express a heartfelt thanks of gratitude to several individuals for their invaluable support and contributions to this project. We would like to express our deepest appreciation to:

Our professors, Dr Purnima Murali Mohan and Dr Peter Loh Kok Keong for their unwavering support, insightful feedback, and continuous encouragement throughout every phase of our research and development.

Mr. Choo Yuh Joo of Starview Technologies, whose expertise, assistance, and valuable insights helped shape the direction and outcome of our project.

Mr Rob Cheney, for his extensive knowledge, helpful advice, and thoughtful recommendations that significantly improved our understanding and approach of the domain.

Their dedication and expertise have helped us achieve our objectives, and we are deeply grateful for their contributions.

CONTENTS PAGES

1. INTRODUCTION

1.1 Objectives

The objective of this project is to develop a Proof-of-Concept (PoC) Artificial Intelligence (AI) model capable of identifying key body parts and analysing golf swings from video using Computer Vision techniques. In addition to the AI model, the project entails building a front-end coaching feedback system. This Front-end system will provide golf coaches with detailed and accurate assessments of their swings, helping them improve their performance by comparing their movements to established benchmarks of good or bad swing mechanics.

1.2 Problem Statement

Currently, it could be more efficient for golfers and their coaches to get immediate, accurate feedback on swing mechanics. Traditional approaches, such as visual assessments by coaches, slow-motion video analysis, and data-collecting wearables, are not only time-consuming but also require substantial coach involvement to interpret results and offer corrective actions (1). This process impedes the ability to make real-time adjustments, undermining the effectiveness of training sessions. There is an urgent need for an advanced, AI-driven system capable of delivering instant, precise feedback and comprehensive swing analyses, aiming to enhance training efficiency and elevate coaching standards.

1.3 Motivation

Building upon the existing challenges faced, the team envisions leveraging cutting-edge computer vision and machine learning technologies to revolutionise the feedback mechanism for swing mechanics. Thereby, empowering golfers with real-time, detailed feedback on their swings, allowing them to understand and refine their technique. Coaches, through the team's AI-driven system, will receive precise effortless analyses of key body points and swing positions, gaining access to comprehensive visualisations and actionable insights, such as detailed angle measurements and positional data via a user-friendly interface. This innovation not only streamlines coaching sessions, elevates the quality of instruction, and allows coaches to focus more on strategic guidance and mentorship. In addition, this breakthrough has the potential to bridge the gap between technology and athletics, transforming the way golfers learn and coaches instruct, ushering in a new era of efficiency, insight, and success on the course.

1.4 Project Domain and Background

The project's domain encompasses the application of highly advanced computer vision technologies to interpret and analyse video data of golf swings. It pinpoints key anatomical features and critical positions during the swing, crucial for mastering the sport. Golfers can conveniently upload their swing videos through an interactive web interface which are then scrutinised by the intelligent AI system, which assesses the recorded movements against recognised benchmarks for efficient golfing techniques. The AI then generates personalised advice for users, providing clear, evidence-based insights to refine their technique and grasp the intricacies of a good swing.

Within this domain, the project harnesses the capabilities of AI to address traditional barriers in golf analytics. These challenges include the subjectivity inherent in human assessments and the difficulty in capturing and interpreting complex movement dynamics. This project hopes to offer a more objective, insightful, and accessible tool for golfers and coaches, promoting a deeper understanding of swing mechanics and enabling targeted improvements in golf skills by utilising computer vision and machine learning capabilities.

2. PROJECT MANAGEMENT

2.1 Project Changes

2.1.1 Transition to MediaPipe

In the initial phase of the project, TensorFlow Pose was being used for golf swing analysis. But after assessing the need for better performance and accuracy, the team felt it should be shifted to MediaPipe, which has better handling of key points and an added advantage in real-time applications.

The decision to switch to MediaPipe was driven by the following factors:

1. Performance Enhancement: MediaPipe offered faster processing times and more accurate pose estimation, crucial for real-time golf swing analysis.

2. Key Point Detection: MediaPipe's ability to precisely detect and track key points in the golf swing provided more reliable data for the model.

3. Ease of Integration: The seamless integration capabilities of MediaPipe with the existing setup minimised disruption and facilitated a smooth transition.

4. Comprehensive Key Points: MediaPipe provides 33 key points compared to TensorFlow's 17 key points, allowing for a more detailed and nuanced analysis of the golf swing.

The transition involved updating the team's GitHub codebase with the MediaPipe Pose Python guide. This update allowed access to pre-built models and drawing utilities for annotating images and videos with pose landmarks. Through this adoption of MediaPipe, the AI model has significantly improved its accuracy and responsiveness.

2.2 Requirements Changes

2.2.1 Meeting Reviews

Every element is critical in ensuring that the AI system delivers accurate, insightful, and practical feedback to golfers of all skill levels. The client meetings' decisions shape the project's development and implementation.

1. Golf Technique – Stack and Tilt Method: The project will focus on refining golf techniques, emphasising the stack-and-tilt method. This approach requires a detailed analysis of various stages of the golf swing, categorised into ten critical checkpoints (P1-P10). Each checkpoint represents a distinct phase of the swing, from the initial stance to the follow-through, serving as markers for evaluating alignment, posture, and movement patterns.

2. Golf Views– Stack and Tilt Method:
    a. Front View:
        i. The focus is on detecting key elements for a stable swing from a front view. The golfer's head should remain steady, with a 90-degree spine alignment at the top to maintain centre. The lower body should stay stable during the backswing and then move towards the target in the downswing. Proper movement of the shoulders and hips during the downswing is crucial for balance and a smooth, consistent swing. This knowledge allows for better accuracy in analysing and improving golf swings.

    b. Down The Line:
        i. For a stable swing "down the line," the spine should be inclined to the ground and remain stable throughout. The tilt of the spine in the follow-through should be consistent, as inexperienced golfers often lack sufficient tilt or experience changes in tilt after the follow-through. Maintaining the swing within a planar V-shaped area ensures good contact with the ball, enhancing overall performance and accuracy.

3. Stick Figure: The stick figure will play a central role in classifying and measuring the golfer's body posture and movements. It will serve as a visual aid and a tool for objective analysis, employing precise

measurement standards to offer insights into the golfer's performance.

4. Data Display Design: The front-end feedback to have a colour-coding system for data points serving as a visual aid to display real-time feedback during the swing

5. Role-based Accessed Web Pages: There will be separate access to the website, one for coaches and one for the students. Coaches will be able to see their students' uploaded golf swings while the students will be able to upload their swings and recommendations from the golf coach.

6. Architecture Design: The client requested research about the best architecture for the solution and in the same to be compared between the client-server model and local software options. If possible, this is the type of modelling the client would like to use, mainly due to its cost-saving benefits.

3. RELATED WORK

3.1 Review Literature and Studies

The concept of using computer vision to improve golf swing sequencing has been explored in various studies, with one of the earliest notable papers being "Visual golf club tracking for enhanced swing analysis" by Gehrig, Lepetit, and Fua, presented at the British Machine Vision Conference (BMVC) in 2003 (2). This paper delved into tracking golf club movements to aid in swing analysis using computer vision techniques.

More recent advancements have been made with the introduction of the GolfDB database and the development of SwingNet, a hybrid deep convolutional and recurrent neural network. This work, published in 2019 by McNally et al., provided a comprehensive video database of golf swings and a model to detect key events in the swing sequence (3). These studies have significantly contributed to the field by offering resources and benchmarks for further research and application in golf swing analysis using computer vision(4).

3.1.1 Current Implementation

3.1.1.1 Golf Swing Sequencing Using Computer Vision

Researchers from Rhodes University have studied the implementation of an automated system for detecting and classifying golf swing events using the GolfDB dataset(5,6). Their approach integrates image processing and machine learning techniques, achieving significant accuracy in identifying swing sequences without relying on expensive equipment like cameras or a motion capture suit. This study represents a pivotal advancement in democratising access to precise swing analysis tools, benefiting golfers of all levels by providing actionable insights for technique improvement.

3.1.1.2 GolfMate

This project tool was developed by Hanyang University which introduces a golf swing analysis system. It integrates pose refinement methodology detecting key poses and refining them(7). This tool offers an efficient approach to giving feedback to golfers and supporting independent training and improvement.

3.1.1.3 Applying Pose Estimation to Predict Amateur Golf Swing Performance Using Edge Processing

This study explores the application of vision-based pose estimation to identify critical moments in the golf swing sequence. This way it eliminates intrusive measuring devices, preserving the natural flow of the golfer's swing and providing rapid and precise feedback directly at the driving range(8).

3.1.2 Current Pose Estimation Models

Pose estimation is a critical task in computer vision that involves predicting the key body joints (key points) in images or videos. This section examines two prominent frameworks for pose estimation:

3.1.2.1 MediaPipe Pose

MediaPipe, an open-source cross-platform framework developed by Google, provides an impressive real-time pose estimation technique. Its pose landmarking model tracks up to a staggering 33 body landmark locations of the various body parts, has a wide range of pre-built models and supports various platforms (9).

3.1.2.2 *TensorFlow Pose*

TensorFlow Pose, a pose estimation framework from TensorFlow machine learning library, employs a variety of deep learning techniques to identify 16 key body joints. It also offers a range of pre-trained models that can be fine-tuned to detect human body poses and is flexible for developers to edit to fit specific use cases (10).

3.2 Key Findings from the Literature

3.2.1 *Golf Swing Sequencing Using Computer Vision*

Researchers explored methods to detect and sequence golf swings using computer vision techniques, focusing on automating the analysis process. Their approach, including an automated golfer detector, significantly enhances accuracy and provides insightful feedback for both golfers and coaches. This advancement aids in understanding swing mechanics and identifying areas for improvement.

3.2.2 *Golfmate*

This project integrates a pose refinement network and explainable golf swing embedding to enhance swing analysis. Their system refines pose estimation, improving accuracy in identifying key poses and offering clear, actionable feedback. The explainable embedding feature enables users to understand swing performance better, facilitating independent skill enhancement and training.

3.2.3 *Applying Pose Estimation to Predict Amateur Golf Swing Performance Using Edge Processing*

This study applies vision-based pose estimation to analyse golf swings, focusing on real-time identification of critical swing moments without intrusive devices. Evaluating posture and swing tempo, the system predicts swing outcomes effectively, offering immediate feedback for continuous performance improvement. Therefore supporting golfers to improve with menial disruption when golfing.

3.2.4 Comparison between Pose Estimation Models

Table 1. Comparison of Models

Feature	MediaPipe Pose	TensorFlow Pose
Framework and Development	Developed by Google, an open-source cross-platform framework	Open-source machine learning framework developed by Tensorflow Community
Ease of use	Easier to deploy for specific use cases due to pre-built models and components	Steeper learning curve but offers greater flexibility and control
Platform Support	Cross-platform (Android, iOS, web, and more)	Cross-platform (with a strong focus on server and cloud environments)
Performance	Optimised for real-time applications on edge devices	Scalable from small to very large models, optimised for high performance on both Central Processing Units (CPUs) and Graphic Processing Units (GPUs)
Customization	Limited to the pipeline components and pre-built models	Highly customizable, with support for custom layers, models, and training loops
Key Landmarks	33 key points	16 key points

Table 1 summarises the comparison of MediaPipe Pose with TensorFlow Pose. In relation to the project AI system, Mediapipe was chosen due to its optimised performance essential for providing immediate feedback during swing analysis as well as the large number of crucial points observed, which ensures model accuracy. Its ease of development aligns well with the project's need for efficient implementation without extensive customization.

3.3 Issues with existing Tools

3.3.1 Manual

The issue with manual analysis is that golf swing analysis currently depends on manual annotation. This involves people identifying and labelling key movements at important stages of a golf swing. Manual analysis is both time-consuming and can lead to mistakes, which leads to inconsistencies and inaccuracies in the dataset. The team utilised a Machine Learning Model (MLM) with the aim of eliminating these issues.

3.3.2 Golf DB

While the GolfDB database holds significant data for a golf swing analysis, it has its limitations. Some areas the database lacks include various swinging styles, no differentiation between players' various skill levels, as well as environmental conditions such as lighting levels. These omissions make the dataset tedious and time-consuming to work with. In addition, it is unable to provide real-time feedback for pre-recorded videos as a training model and by extension, unable to provide real-time feedback for the user.

4. METHODOLOGY

4.1 System Architecture

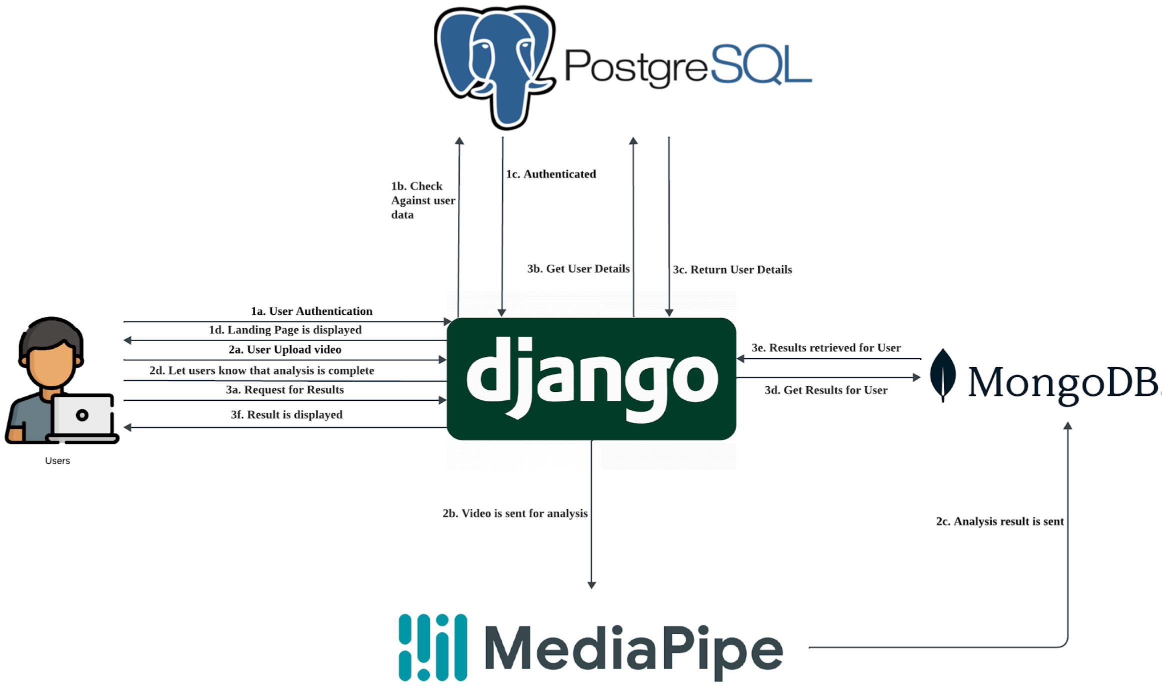


Figure. 1 System Architecture

The system's workflow in Figure 1 initiates when a user uploads a video to the Django web application interface. Once uploaded, the video is sent for analysis to the MediaPipe backend, which utilises a machine learning model trained to analyse golf swings. The results of the analysis are then returned to the web application after it has been processed and stored in a MongoDB database to ensure long-term reliability and scalability. Furthermore, PostgreSQL will manage all user-related data concurrently, saving and retrieving relevant details as needed. Overall, this architecture will help optimise the integration of user interaction, data processing and storage functionalities across distinct technologies, allowing for efficient handling of video uploads and analysis within the system.

4.2 Use Case/Storyboard

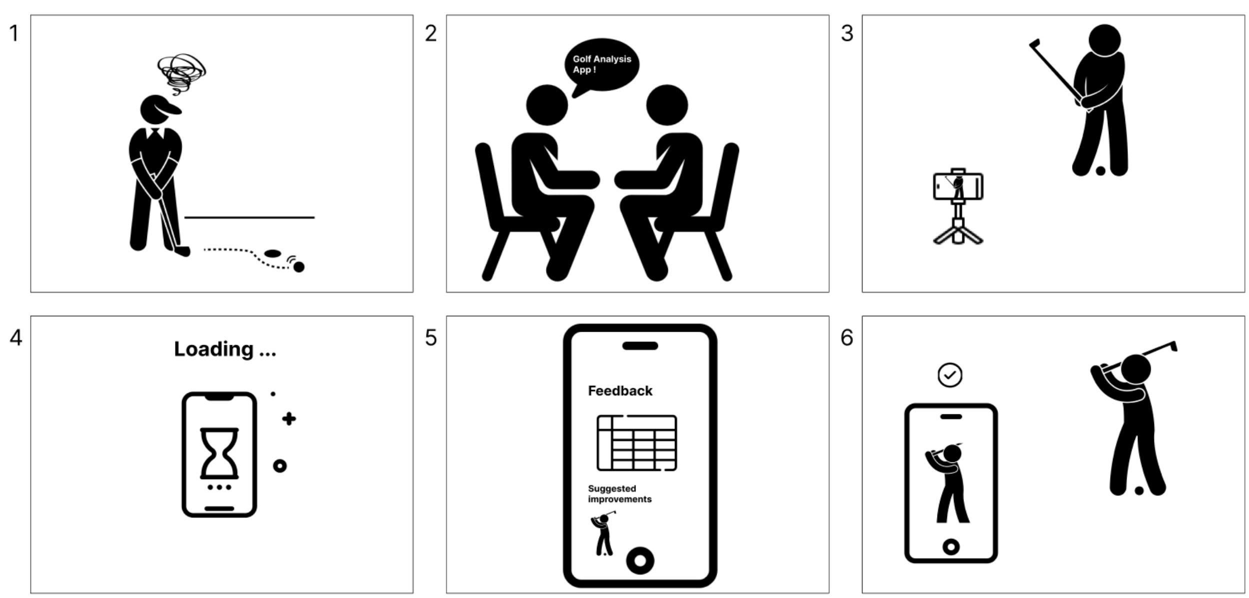


Figure. 2. Use Case Diagram

The Storyboard introduces the interaction and motivation of the end users with the application shown in Figure 2. In the first frame, Alex is frustrated at his golf games due to his poor drives leading to a bad score. In the second frame, Alex's friend John, who he plays golf with, introduces him to the Golf Swing Analysis application. Next Alex uses the application and films his golf swings and then the application will take in the data from the video and process it. The application will then show him feedback on his swings at the different positions with suggested improvements. Alex then recognizes his errors and corrects his swing according to the suggestions which helps Alex improve in his golf game.

4.3 Frontend Development

4.3.1 Tools

4.3.1.1 Figma

Figma was chosen as the primary design tool for its comprehensive capabilities in creating and sharing design wireframes. By using Figma, the team can efficiently design wireframes that serve as detailed blueprints for the development process. This visual representation is crucial for presenting to clients, allowing them to see and understand the proposed designs clearly. This ensures that client expectations are aligned with the project vision from the outset. Additionally, Figma enhances collaboration within the frontend team by providing a clear visual guide, reducing miscommunications, and ensuring everyone is on the same page regarding design direction.

4.3.2 Framework and libraries

4.3.2.1 Tailwind: Utility-first Cascading Style Sheets (CSS) framework for styling

Tailwind was chosen for styling and design implementation, with its utility-first CSS framework, which has a lot more to offer compared to conventional CSS. The team prefers it due to the intuitive and productive approach towards styling that makes the whole process easy and productive. The team is enabled to avoid complexities and maintenance challenges attached to traditional CSS files through the use of Tailwind. This streamlined approach not only accelerates development but also ensures a cleaner and more manageable codebase, ultimately leading to a more efficient workflow and faster project completion.

4.3.2.2 FontAwesome

FontAwesome was integrated into the project for its extensive and versatile icon library. This tool provides a comprehensive set of icons that can be easily incorporated into the application, enhancing its visual appeal and user interface consistency. The ease of integrating FontAwesome into the project saves time and effort for the development team, allowing them to focus on other critical aspects of the application. The consistent use of icons from a single library also ensures a uniform look and feel across the application, contributing to a polished and professional final product.

4.3.3 Client Communication and Feedback

Effective collaboration with the client was a key aspect of the team's development process, ensuring that the final product met their expectations and requirements. The team began by thoroughly discussing the client's needs, goals, and vision in several initial meetings to gather detailed requirements and set clear expectations (see Appendix A for a summary of these conversations). Using Figma, the team developed detailed wireframes that visually represented the proposed solution that will be discussed further in Section 5.2. After presenting these wireframes to the client, the team gathered their feedback and made necessary adjustments in an iterative process. Throughout the development, the team maintained regular communication, providing updates and seeking feedback to ensure alignment with the client's vision. Regular review sessions allowed the client to see the project's progress and provide additional input, leading to final adjustments and refinements. This collaborative approach ensured the client was actively involved throughout, resulting in a final product that met their expectations and requirements.

4.3.4 Design Principles Applied

Primarily, the design of the front end looked at improving the user experience, making it simple, intuitive, and engaging. All this was done after intense user research to know what the target users needed, preferred, and their pain points. This informed the way user personas and journey maps were created, thus guiding the process to ensure seamless and logical user flows.

4.3.4.1 Key UX strategies implemented

1. Consistent Navigation: Navigation elements are standardised across the site, making it easy for users to locate information and understand the structure of the website.

2. Visual Hierarchy: Important information and primary actions are prominently displayed, guiding users' attention and improving content discoverability.

3. Feedback Mechanisms: Interactive elements provide immediate feedback (e.g., hover effects, loading indicators) to enhance user confidence and satisfaction.

4.3.4.2 Key User Interface (UI) Design Standards implemented

1. Typography: A hierarchy of fonts and text styles ensures readability and highlights key content. Headings are distinguishable from body text, enhancing content structure.

2. Colour Scheme: The chosen colour palette aligns with the brand's identity and is used consistently across all components, aiding in brand recognition and visual cohesion.

3. Iconography and Imagery: Icons and images are used purposefully to support content, improve comprehension, and break up text-heavy sections.

4.4 Backend Development

4.4.1 Machine Learning

The group will examine the key steps in the machine learning process in this section, which include training, building models, and data preprocessing. To create a reliable and accurate pose estimation classification model, these steps are essential. Each process will be covered in detail by the team, along with the methods and strategies used to properly prepare the data, define the model architecture, and train the model. The team will evaluate the trained model's performance and capability for generalisation during the model evaluation phase, which is covered in Section 5.1.

4.4.2 Data Preprocessing

Data Preprocessing is a crucial step in preparing data needed for training a pose estimation classification model. This process ensures the input data is in the correct format and a sufficient quantity of quality data to make it suitable for the specific machine learning task. The key steps in data preprocessing for pose estimation classification include data collection, data extraction, data augmentation, and landmark detection.

4.4.2.1 Data Collection

The first step in Data Preprocessing consists of gathering data from various sources. The data collected for training testing and validation are taken from data provided by the Client, videos uploaded on to YouTube and images from various websites.

4.4.2.2 Video Frame Extraction

This step consists of extracting the raw data gathered which is in the form of videos and converting them to a format suitable for the input data type for training of the machine learning model. The model chosen is a pose estimation classification model that takes images as input and outputs a classification based on the prediction of the input image given. The team created a script to convert a video with formats of either mp4, avi or mov to a folder of frames from the video. Refer to Figure B1 under Appendix B for the code snippet.

4.4.2.3 Data Augmentation

The next step in data preprocessing is data augmentation. Data augmentation is the process of artificially generating new data from existing data to overcome class imbalance and overfitting problems in the golf analysing model as the data set for certain positions is too small. The video frames extracted are manually sorted into folders for each position. Due to the speed of golf swings in general, some positions have only a few frames collected for each video compared to other positions. To combat this class imbalance, the original image is augmented by applying the following augmentation techniques:

1. **Width and height shifts**: Randomly shifts the image horizontally and vertically within the specified range
2. **Shear Transformation**: Applies a shear transformation to the image
3. **Zoom**: Randomly zooms into the image
4. **Horizontal Flip**: Randomly flips the image horizontally
5. **Fill Mode**: Determines how to fill newly created pixels after a transformation

```
# Data augmentation settings
datagen = ImageDataGenerator(
    width_shift_range=0.1,  #
    height_shift_range=0.1,  #
    shear_range=0.2,  # Shear
    zoom_range=0.1,  # Zoom ra
    horizontal_flip=True,  # R
    fill_mode='nearest'  # Poi
)
```

Figure. 3. Code Snippet for Data Augmentation settings

The augmented image as shown on the right in Figure 4 is horizontally flipped with the image being shifted down vertically. The data augmentation process is repeated 5 times for each image, expanding the total dataset by 400%.

Figure. 4. Original (Left) vs Augmented (Right) Image from the Real Golf Coach involved in the Project

4.4.2.4 Landmark Detection

The final step in the team's data preprocessing flow is landmark detection. This step involves identifying key body landmarks in images and formatting the data for model input. Landmark detection is a crucial step that ensures the model can accurately learn and classify different poses based on the positions of key body landmarks. Performing this step enhances the model's ability to generalise and perform well on various poses and conditions. Refer to Figure B2 under Appendix B for the Code Snippet for saving landmarks detected into a CSV (comma-separated values) file.

There are **33** landmark**s** for key points shown in Figure 5 below. The pose landmarks are derived by utilising Google's MediaPipe library. The library has a pose landmarker task that helps in detecting landmarks of human bodies in an image or video through machine learning models. The landmark results are then saved into a CSV file.

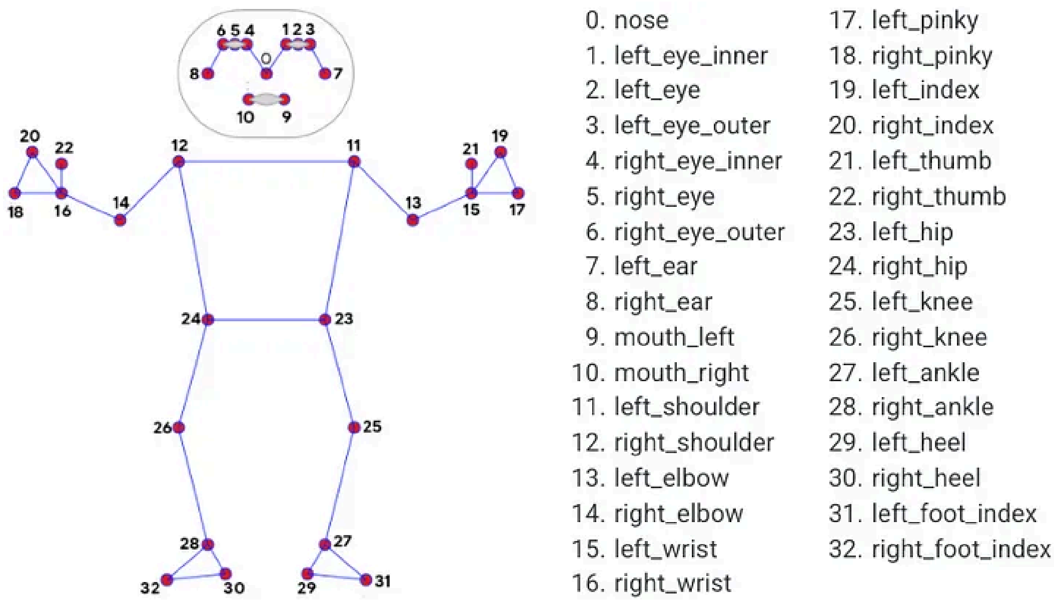


Figure. 5. 33 pose landmarks(9)

4.4.3 Model Building and Training

Model Building and Training are the next steps after preprocessing data in the machine learning process. For a pose estimation classification model, these faces involve defining the model architecture, compiling the model, and training it using labelled data. This section outlines the processes of model building and training, explaining their significance and the steps involved.

4.4.3.1 Model Building

The model architecture is defined using a sequence of layers, where each layer performs specific computations. The input layer size corresponds to the number of features in the dataset, hidden layers apply transformations to capture patterns, and the output layer size corresponds to the number of classes. A model compiler with optimizer, loss function and metrics is defined to ensure the model can learn from the data while minimising errors and improving its performance over time. Refer to Figure B3 under Appendix B for the Model Architecture Code Snippet.

4.4.3.2 Model Training

This step uses the model defined earlier, trains it using the training data and validates its performance on the validation data. The training process involves adding a checkpoint callback which stores the checkpoint that has the highest validation accuracy. Early stopping is enabled to ensure the model stops training once the model stops improving its accuracy after a certain threshold is met. The training process iteratively learns from the training data while simultaneously validating its performance on unseen data. This process guided by specified parameters and callbacks helps in optimising the model's weights, ensuring it generalises well and performs accurately on new data. Refer to Figure B4 under Appendix B for the Model Training Code Snippet.

4.4.3.3 Model Training History

The training metrics are also plotted on a graph for visualisation of the accuracy and loss for the concurrent training and validation of the model. Refer to Figure B5 under Appendix B for the Model Training History Code Snippet. The chart in Figure 6 shows the accuracy and loss scores from 0 to 1 for the 70 epochs during model training. Accuracy is the percentage of correct classifications that the model achieves, hence a higher score reflects a better ability to predict correctly the classifications. Having a higher accuracy score out of 1 is ideal. Loss, on the other hand, measures how well or badly the model is doing. If there are more errors, the loss will be higher, meaning that the model did not perform well. Hence achieving a lower loss is ideal. In short, the model's training performed well as shown below. The accuracy of both training and validation steadily increased while the loss score for both steadily declined.

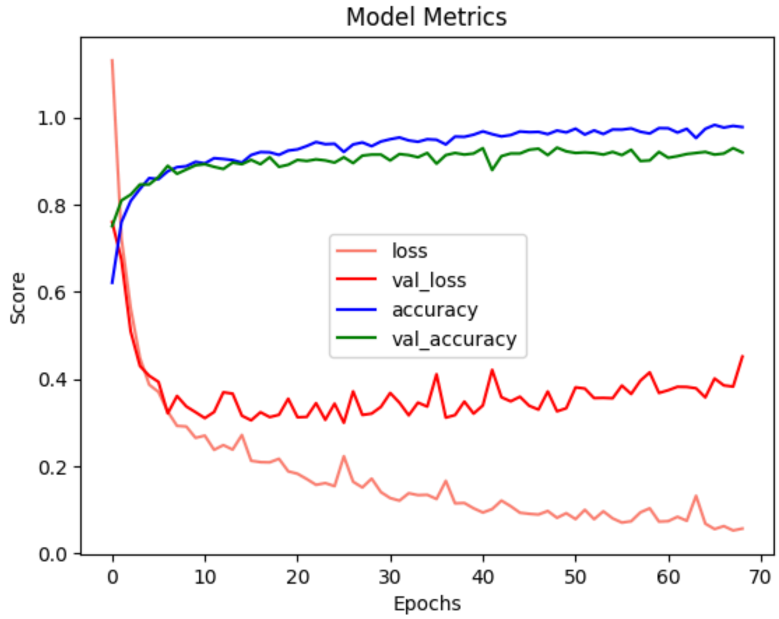


Figure. 6. Chart of the Model's metrics throughout training phase

A confusion Matrix is also plotted to visualise the number of true positives for each golf position in Figure 7 . Refer to Figure B6 under Appendix B for the Code Snippet for Plotting the confusion matrix. The model performs well in classes P1, P8, P9 and P10, with high true positive counts and minimal misclassifications. Classes P4 and P6 show moderate performance but also exhibit misinterpretations with classes such as P5 and P3 respectively. Classes P2, P3, P5 and P7 display more significant confusion, indicating that the classes might have overlapping keypoint features that the model may find difficult to distinguish.

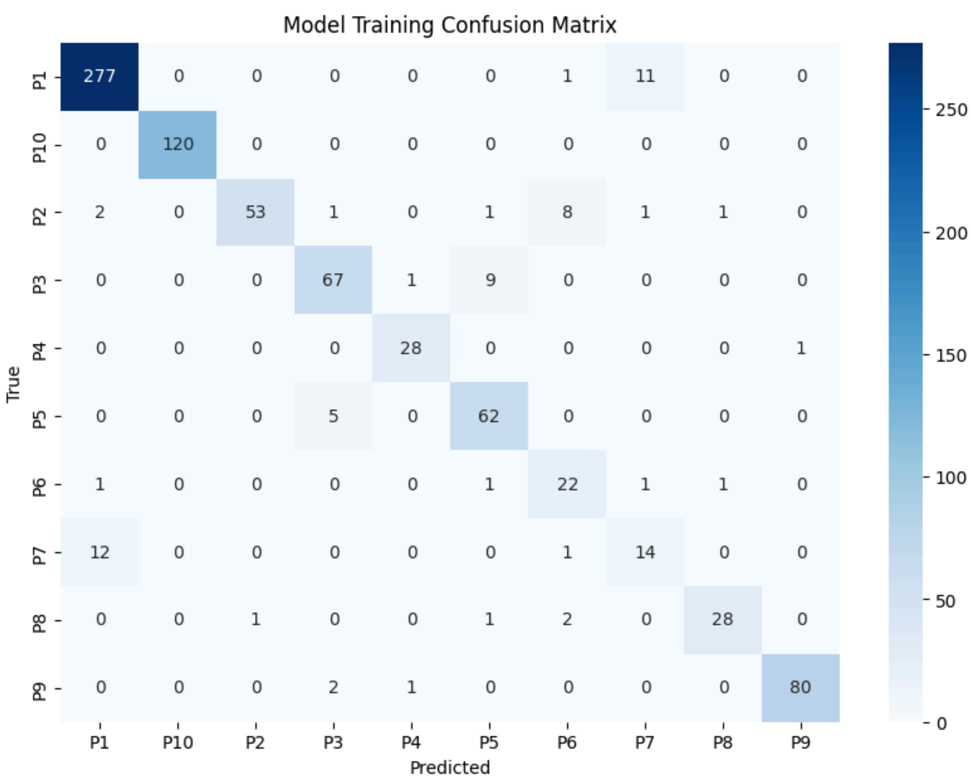


Figure 7. Confusion Matrix visualising Model accuracy

4.4.4 Model Inference

Once a model has been trained, the final step in machine learning is to utilise the trained model to make predictions on new, unseen data such as test images, videos, or live feeds from a webcam. This section delves into the processes involved in utilising the model for prediction, highlighting their importance and detailing the necessary steps

4.4.4.1 Initialization and Setup

Firstly, the script begins by importing necessary libraries and setting up configurations using “argparse”. It accepts arguments such as the model path (-m), confidence threshold (-c), and input source (--source), which can be either an image or a video. Following this, it utilises MediaPipe library to detect landmarks on human poses and loads a pre-trained Keras model for pose classification based on these landmarks.

4.4.4.2 Confidence Threshold

A Confidence Score is a number between 0 and 1 that shows the probability that the output of a Machine Learning model is correct. Each prediction from the model has a Confidence Score, the higher it is, the more confident the prediction is accurate(11).

Table 2. Typical confidence associated with a given score

Score Value(%)	Score Meaning
90 - 100	A near exact match
> 70	High confidence - typically a good answer
50 - 70	Medium confidence - typically a fairly good answer
30 - 50	Low confidence - typically a related answer
< 30	Very low confidence - typically does not answer
0	No match, so the answer is not returned.

Setting a confidence threshold parameter filters out low-confidence pose detections, and ensures that only high-quality predictions are considered. Thereby, reducing false positives and maintaining the reliability of the predictions.

4.4.4.3 Preprocessing Steps

Once the model loading process begins, it makes use of preprocessing techniques for brightness adjustment and noise reduction to improve pose detection accuracy. Adjusting the brightness from the input source can help to ensure consistent lighting conditions making it easier to accurately detect landmarks regardless of lighting variations. Furthermore, reducing the noise from the input can remove unnecessary artefacts to have a clearer image.

4.4.4.4 Processing Images and Videos

When the user inputs a source for either an image or a video, it first checks and distinguishes between them. For image processing, landmarks are detected and pose classification is predicted using the loaded Keras model. Then the detected poses are visualised drawing the landmarks connection on the image and displaying the predicted pose class with the skeleton model in the image. Similarly to image processing, video processing first captures the frames from the input source. Each frame is then used for landmarks and pose classification is predicted in real-time, while also visualising the connections and displaying the predicted pose class with the skeleton model.

4.4.4.5 Data Analysis

Using the pre-trained model, this can efficiently detect human poses and classify them into predefined pose classes. It also conducts data analysis for calculating various angles, useful for coaches. One key angle calculated is the shoulder tilt. This refers to the angle at which the shoulders are tilted during the swing and can be computed using the coordinates of shoulder landmarks detected by MediaPipe shown in Figure 8 below. This is how the angle is calculated by using the following formula(12):

$$\vec{AB} = \theta_1 = arctan2(ay - by, axe - bx)$$
$$\vec{BC} = \theta_2 = arctan2(cy - by, cx - bx)$$
$$\theta = \theta_2 - \theta_1$$
$$\text{angle} = \theta \times 180/\pi$$

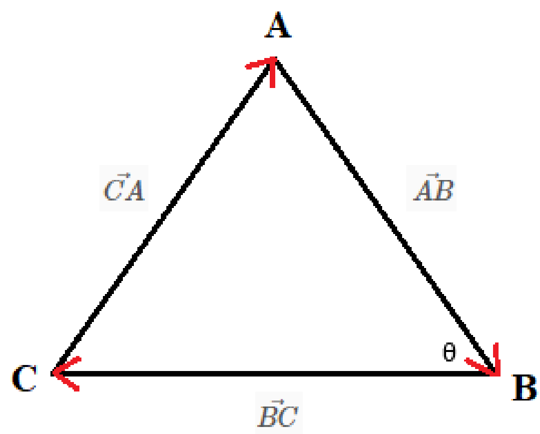


Figure. 8. Vectors visualisation from triangle ABC

Additionally, the team has computed average shoulder tilt angles for each pose class, visualises their distributions, and provides insights into the variability and typical ranges of shoulder tilts across different poses. These findings are discussed in detail in Section 5.1.

4.4.4.6 Error Handling

The script includes mechanisms to handle potential errors or edge cases. If certain landmarks are not detected, the script skips the angle calculation for that frame. Predictions with confidence scores below the threshold are discarded to maintain high prediction quality. The script uses try-except blocks to catch and handle unexpected errors gracefully, ensuring continuous execution without abrupt crashes.

4.5 Application of Concepts taught

4.5.1 Project Management

The team adopted a hybrid approach for the Integrative Team Project, blending elements of both Waterfall and Agile methodologies to leverage the strengths of each while mitigating their limitations. Initially, the Waterfall methodology was employed for comprehensive planning and establishing clear project phases, ensuring thorough documentation of project objectives, requirements, and design specifications. This structured approach facilitated the delivery of project requirements at defined milestones. However, acknowledging the Waterfall model's inflexibility regarding changes, the team incorporated Agile methodology for the development process, allowing for high flexibility in adjusting features and accommodating client feedback. This combination ensured the rigidity needed for initial planning and documentation, coupled with the agility required for a dynamic and adaptable development cycle, thereby preserving the best aspects of both methodologies.

4.5.2 Web Development

The team followed the RESTful principles to ensure that the Application Programming Interface (API) was designed efficiently and securely. This facilitated seamless integration between the front end and the back end components of the web application, which enhanced the functionality and user experiences. Following these principles, the team developed a robust and scalable API that supports the interactive and dynamic nature of the web application.

4.5.3 Database Management

The team utilised relational tables to efficiently organise user-related information, ensuring data integrity and facilitating quick access. Additionally, recognizing the need for flexibility in storing unstructured data related to golf swing models and analyses, thus integrated NoSQL databases. This choice optimised performance for large-scale data operations, demonstrating the ability to adapt database technologies based on the requirements of the data at hand.

4.6 New Concepts learned and their application

4.6.1 Artificial Intelligence

4.6.1.1 Machine Learning

The team began training machine learning algorithms with a wide range of datasets containing various golf swing techniques. The algorithms were fine-tuned through an exhaustive method of interactive learning and optimization to detect and recognize intricate patterns and relationships within the data. This refinement, also known as trial and error, allowed the algorithm to generate accurate predictions about the quality and the characteristics of golf swings. As a result, these advancements provided the team with valuable insights that helped to improve golfers' performances.

4.6.1.2 Computer Vision

The team explored various new techniques in Computer Vision to process and analyse video footage of golf swings. The team learned to split the video into individual frames and apply algorithms to detect the golfer and track their movements throughout the swing. These techniques allow for a detailed examination of each golf swing, providing insights into mechanics and performance.

4.6.1.3 Pose Estimation

The team also delved into Pose Estimation techniques to identify and track key body joints of the golfer throughout the swing. Usage of pose estimation provides precise data on body alignment and movement patterns, which are crucial for evaluating swing techniques. This can help to extract detailed information about the golfer's joint positions and movements and offer targeted advice to golfers, aiming to improve their swing efficiency and overall performance

4.6.2 Data Science and Analytics

4.6.2.1 Data Preprocessing

The team recognized the importance of data preprocessing. This initial phase ensures that the data is devoid of inconsistencies, errors and noise. Addressing these issues guarantees that the data used in the subsequent stages was reliable, consistent, and conducive to effective analysis.

4.6.2.2 Feature Engineering

Following the preprocessing stage, the team crafted new features from the raw data, capturing intricate patterns and relationships that were previously overlooked. This enriched the dataset and also improved the accuracy and reliability of the models.

4.6.2.3 Performance Metrics

The evaluation of the machine learning model was conducted using a suite of performance metrics like the precision, recall and f1 score. These metrics are essential for understanding the model's accuracy and its ability to minimise false positives and false negatives. The team iteratively refined the model based on these evaluations, ensuring optimal performance and a balanced approach to accuracy. This involved calculating these metrics, closely monitoring changes throughout the development phases and adjusting the model accordingly. This process allowed the gain of valuable insights into the model's strengths and weaknesses, guiding further development.

4.6.3 Frontend development process

4.6.3.1 Tailwind

Using Tailwind CSS has improved the styling process a lot, and its utility-first approach greatly simplified design handling in all the team's projects. Notably, the predefined classes in Tailwind have permitted the team to style directly within the Hypertext Markup Language (HTML), resulting in drastically reduced files of traditional CSS and eliminating back-and-forths in different files. The process became further swift, and the application's design became more consistent. Not only that, but the responsive utility classes in Tailwind also enabled a flawless user interface flow on any screen or gadget without having to integrate strenuous media queries. Then Tailwind's customizability easily allowed the team to make it work best for the project at hand, with optimised flexibility without much time wastage. In a word, Tailwind styling is efficient, maintainable, and scalable.

4.6.3.2 Requirement changes and iterative changes

Changes in requirements and the process of iterative development have further enhanced the role of flexibility and communication in the team's project management practices. First, the team would gather detailed requirements with the client; this gives a clear guideline on the path to be taken with the project. However, some changes were necessary due to new findings and feedback from the project. Moving on an iterative development approach helped the team to take these amendments easily and meanwhile make continuous development that was in tune with the client's needs. Regular check-ins with the client and updates have been a key factor in reviewing progress, gathering feedback, and making necessary adjustments. This iterative process not only keeps the project on track but also improves client satisfaction by ensuring their vision is continuously reflected in the development.

4.6.4 Collaboration and team communication

Effective collaboration and team communication have been pivotal in navigating the complexities of the ongoing project. By fostering an environment of open communication, the team ensures that every team member is aligned with the project goals and aware of any changes or updates. Regular team meetings facilitate real-time communication and quick resolution of issues. Collaborative platforms such as Figma allow both the design and development teams to work in sync, ensuring that the visual and functional aspects of the project are cohesively integrated. This collaborative approach enhances the efficiency and productivity of the team and ensures that the project benefits from diverse perspectives and expertise, leading to a more refined and successful ongoing development process.

5. RESULTS

5.1 Testing (System/User)

5.1.1 Model Evaluation and testing on Image Input

The Model's accuracy is tested on a validation dataset consisting of images from the 10 different poses (P1 to P10). The data given for validation is not used in the training of the model and is taken from TheDIYGolfers Website(13). The validation dataset consists of 5 images for each class and the results of the validation can be seen in Figure 9 below. Overall, the model achieved an overall accuracy of 68% indicating that the majority of classifications were predicted accurately. Classes P1, P2, P3, P6, P8 and P9 have high precision, recall, f1-score and support which indicates the model performs well for these classes. However, Classes P4 and P10 have a score of 0 for precision, recall and f1-score which indicates that the model performs badly on these classes. This could be due to the lack of sufficient data for these classes or the model's inability to distinguish these poses apart from the other poses. Class P9 has a low precision of 0.59 but a high recall of 1.00 resulting in a moderate f1-score of 0.73. This indicates that the model can identify all instances of P9 but identifies other poses as P9.

Classification Report:

	precision	recall	f1-score	support
P1	1.00	1.00	1.00	5
P10	0.00	0.00	0.00	3
P2	0.60	1.00	0.75	3
P3	0.67	1.00	0.80	4
P4	0.00	0.00	0.00	2
P5	0.80	0.80	0.80	5
P6	1.00	0.50	0.67	2
P7	0.33	0.20	0.25	5
P8	0.67	0.80	0.73	5
P9	0.57	1.00	0.73	4
accuracy			0.68	38
macro avg	0.56	0.63	0.57	38
weighted avg	0.60	0.68	0.62	38

Figure. 9. Validation report on model

The Validation report above presents the evaluation results of the model. In addition, a confusion matrix is plotted for visualising the predictions against the ground truth in Figure 10 below. The figure shows the misclassification of P4 and P5 resulting in high recall but low precision as explained earlier. P7 is also classified wrongly for 4 out of 5 images whereas P10 is also classified 3 out of 3 predictions as P9 with the other 2 images classified as unknown poses.

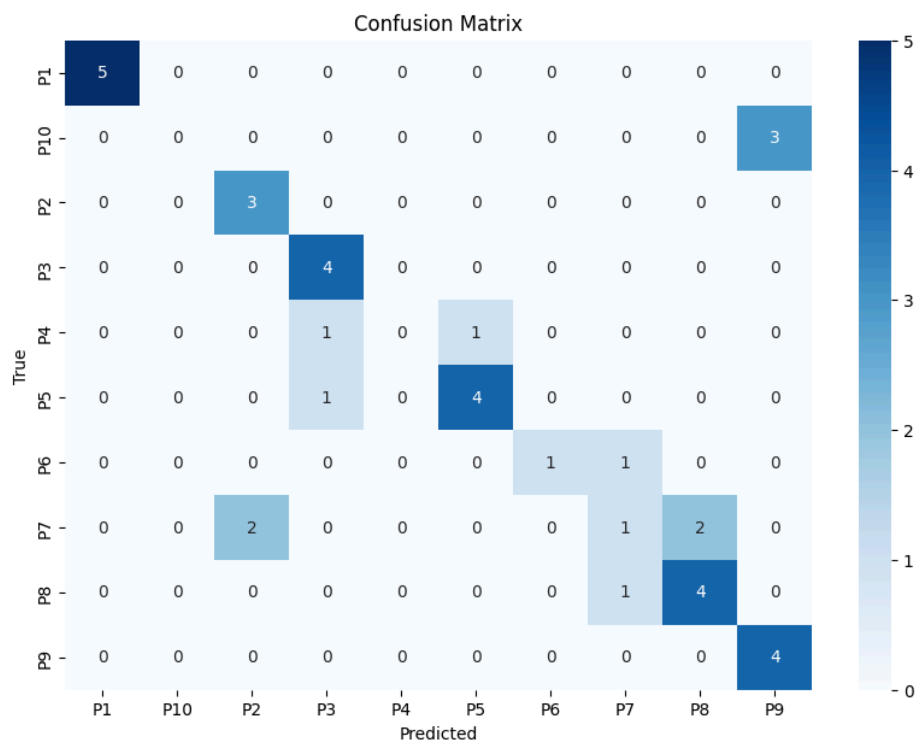


Figure. 10. Validation Confusion Matrix

The Figure 11 shown below, the validation results for each position with the True label being the actual class and the Predicted label being the predicted class. Green colour signifies that the position is predicted correctly whereas red on the other hand signifies a wrong prediction.

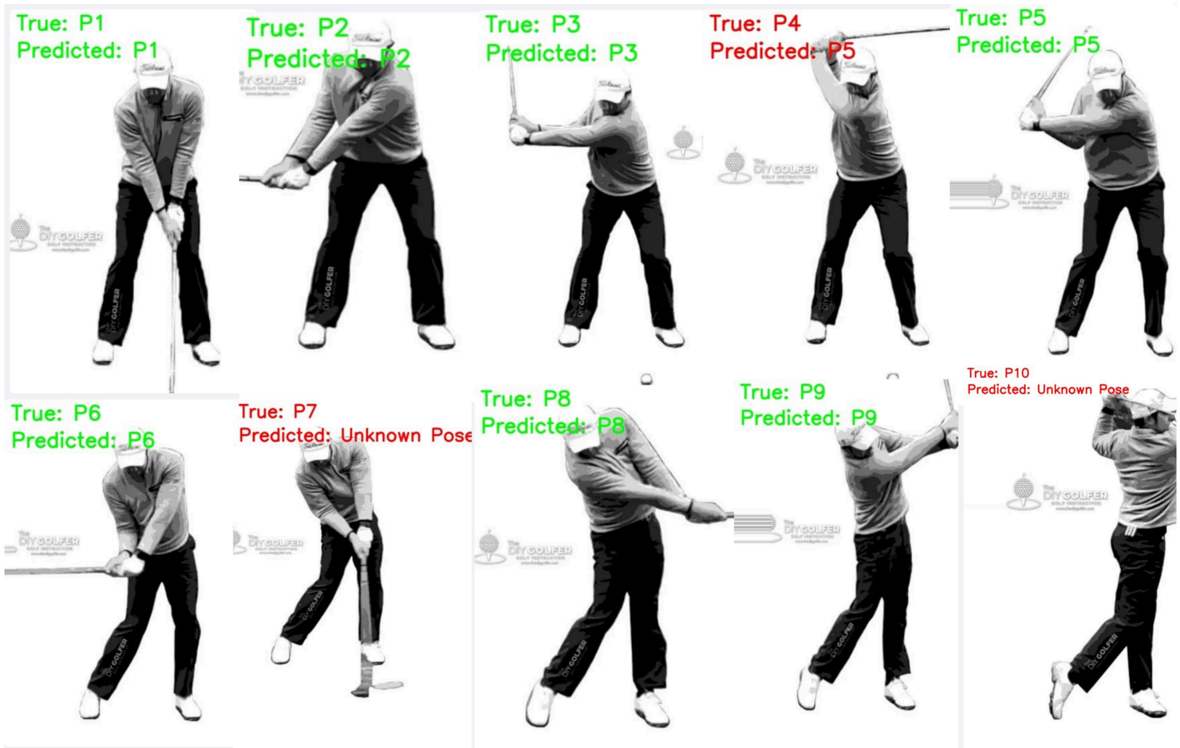


Figure. 11. Model Prediction on validation dataset

5.1.2 Model Inference on Video Input

The team tested the model's inference on new and unseen data in the form of video input. The shoulder tilt angles are taken for each position and used to plot a chart using Matplotlib's pyplot. The team plotted a normal distribution curve centred around the ideal shoulder tilt angle with a standard deviation calculated from the prediction results. This curve visually represents the expected spread of angles if they were normally distributed around the ideal value. By comparing the predicted shoulder tilt angles with the ideal values, the golf coach can assess the accuracy and consistency of the user's performance. For each pose class, the team generated a plot with the following key elements:

Table 3. Shoulder Tilt Angle Distribution Analysis

Elements	Visual Representations
Normal Distribution Curve	Plotted in green with an area fill to show the probability density.
Predicted Angles	Red vertical lines represent each prediction.
Ideal Angle	A blue line for the ideal value.
±1 Standard Deviation	Orange lines for the range within one standard deviation of the ideal angle

A sample of the prediction outcome can be seen in Figure 12 below. The chart is shown after running the model inference script on a video of a golf swing sequence. In most classes of poses, the predicted angles are quite accurate to their corresponding ideal values. Consistency: The low standard deviations across pose classes of the predicted pose angles suggest that the model's predictions are consistent. It should be accounted for why P4 does not have detections, which may be pointing at some issue with the model in detecting this pose or a specific data collection condition. All in all, the model does well in predicting shoulder tilt angles, in that most predictions are aligned quite close to ideal values and one standard deviation around them. In this way, the model is not just as accurate but also very consistent. This raises an area of exception for further exploration of the P4 pose. Refer to Figure B7 under Appendix B for the code snippet for plotting the sample model prediction result.

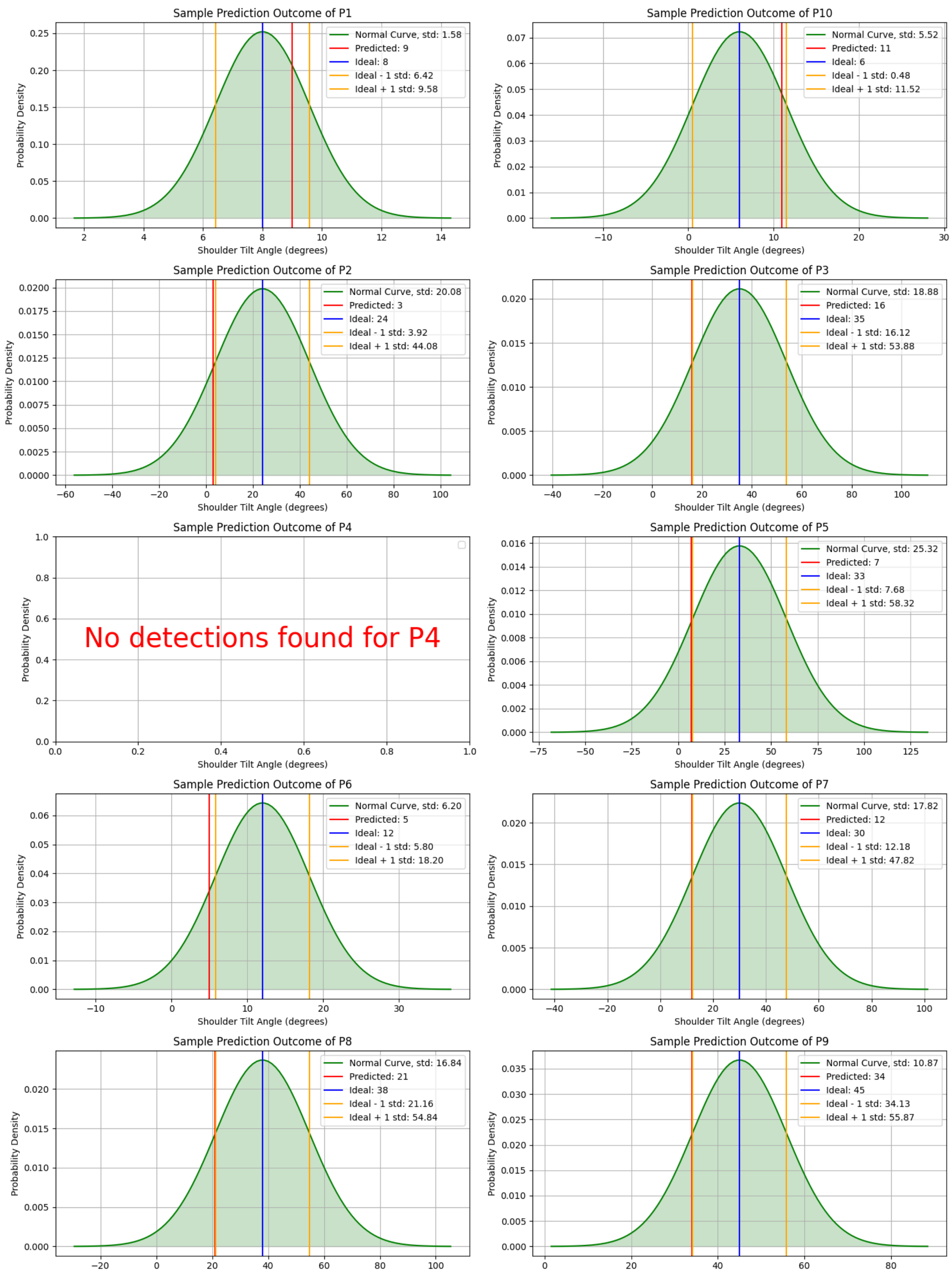


Figure. 12. Sample Model Prediction Result on Original Video

5.2 Frontend Development

5.2.1 Comparison Between Wireframe and Developed Frontend

The front end developed by the team, as shown in Figure 13 to 24 below, will closely represent the wireframe that was first agreed upon by the client. All critical elements of layout structure, content hierarchy, and navigation pathways are retained with the most careful attention so that they follow the client's expectations of the letter. That standard makes the design remain uniform to pre-established standards and to what the client envisioned for the project, hence making it a seamless segue into its design and development.

Table 4. Frontend comparison

Wireframe	Developed Frontend

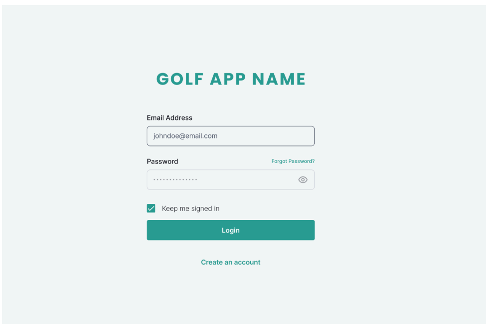


Figure. 13. Wireframe Login

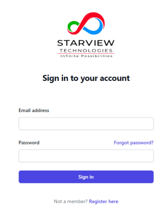


Figure. 14. Frontend Login

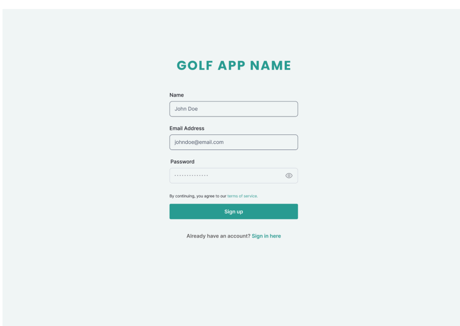


Figure. 15. Wireframe Login

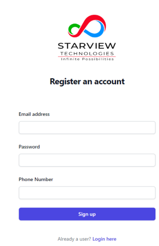


Figure. 16. Frontend Login

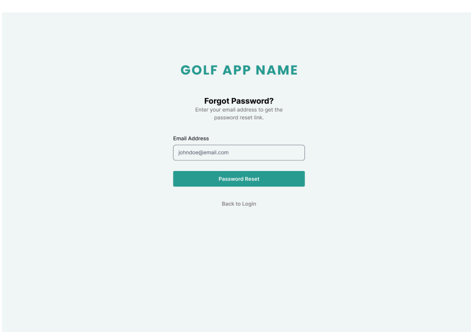


Figure. 17. Wireframe Login

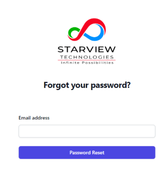
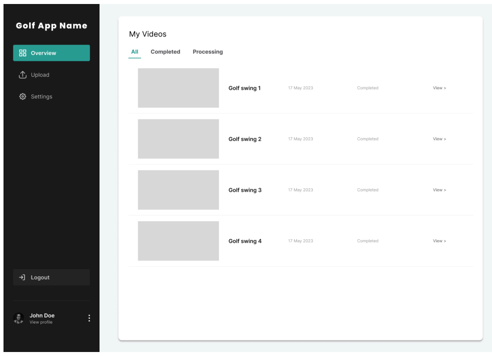


Figure. 18. Frontend Login



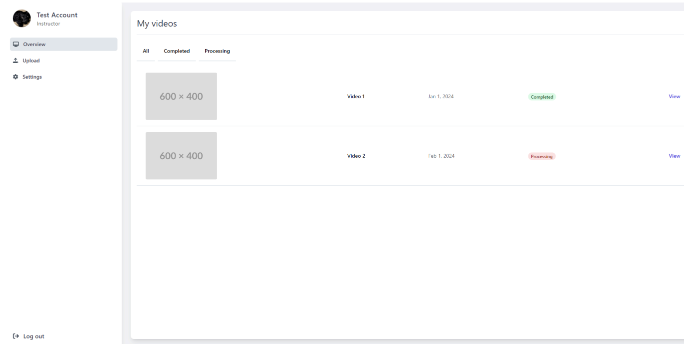
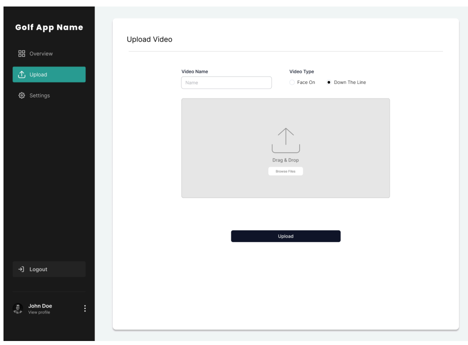


Figure. 19. Wireframe Login

Figure. 20. Frontend Login



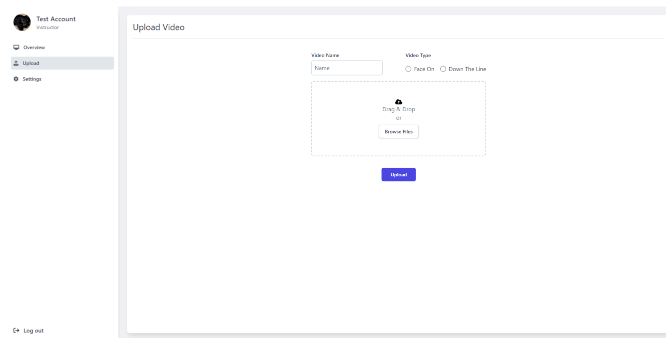
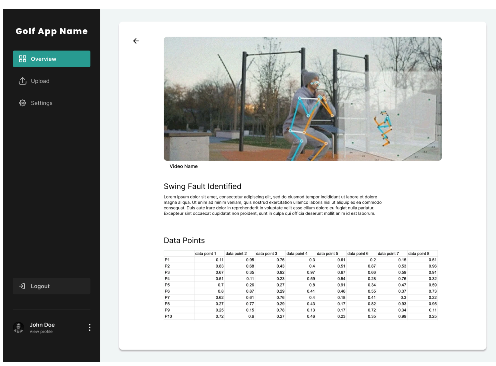


Figure. 21. Wireframe Login

Figure. 22. Frontend Login



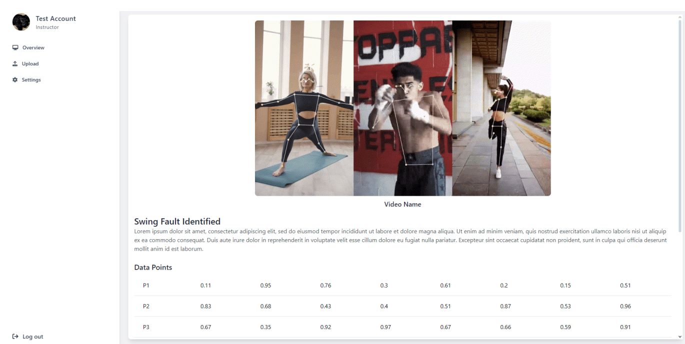


Figure. 23. Wireframe Login

Figure. 24. Frontend Login

6. DISCUSSION

6.1 Analysis of Results

6.1.1 Testing and Model Evaluation Results Analysis

This model evaluation and testing involved checking the efficiency of a machine learning model on a dataset simulating validation data with images representing different golf poses. The accuracy reached by the model is 68%, meaning that most cases are correctly predicted, but results really fluctuate in different classes:

1. High-Performance Classes: Classes P1, P2, P3, P6, P8, and P9 demonstrated high precision, recall, f1-score, and support, which means that the model distinguished these poses quite effectively. High precision and recall values for these classes indicate that the model had low occurrences of false positives and false negatives for these classes.
2. Poor-performance Classes: Classes P4 and P10 received a score of 0 for precision, recall, and F1 score; that is, the model did not make any proper classification of instances for these two poses. Such kind of poor performance might be due to either inadequate training data in these poses or mostly due to the inherent difficulties of distinguishing these from the other poses.

Class P9 is an irregular case, whereby precision amounted to 0.59 and recall to 1.00—resulting in a f1-score of 0.73. This, in other words, means that while this model had high recall, it also misclassified many poses as P9 because of low precision.

The latter was also validated by a confusion matrix representing the classification results visually, in which specific misclassifications could be singled out. For example, P4 and P10 were constantly confused, with P10 predominantly misclassified as P9. This pattern of misclassification may indicate a systemic issue with the model's sensitivity to differentiate between specific poses.

6.1.2 Model Inference Analysis with Video Input

The ability of the model to make inferences was further explored by the use of video input data, which was the tilt of shoulders in the act of a golf swing. The distribution of these angles overlaid with normal centred around an ideal value is shown:

1. Accuracy and Consistency: For most of the pose classes, the model's prediction was near ideal values, and low standard deviations indicated consistency in the performance across multiple predictions.
2. Exception for P4: It is important to point out that P4 had no detections, which could indicate a problem with either the model's ability to recognize this pose or in the data collection process.

In the visualisation of results, the normal distribution curves for each pose were done, with the predicted angles, ideal angles, and their standard deviations clearly denoted. This gave a better view of the model's performance in particular related to the accuracy and consistency of recognizing shoulder tilt.

The test exposed both the strengths and areas of improvement in the model. The results exhibit accuracy in most of the classes, but individual poses like P4 and P10 should be detected more accurately. It is preferable to increase the number of samples for both these classes in the training set or work on the model architecture improvement. Further, more information related to shoulder tilt angles can be helpful in drawing practical conclusions or making applications like sports coaching. Other refinements in the model could be achieved through further exploration and perhaps better classifying underperforming classes.

6.2 Limitations

6.2.1 Lack of Comprehensive Datasets

One of the primary challenges faced during the development of the project's AI system was the availability and quality of the datasets. The system required substantial amounts of video footage and data to train the machine-learning model effectively. Unfortunately, the team struggled to find sufficient golf-specific datasets, which hindered the team's progress. Additionally, reliance on coaches to provide adequate video footage of their swings further delayed the data collection process. As a result, this dependency delayed the data collection process essential for training the AI model.

6.2.2 Computational Resources and Processing Time
Another significant limitation involved the computational resources necessary to process and analyse the extensive data collected. Implementing complex AI algorithms on high-resolution video and motion capture data demanded considerable computing power and memory. As a result, processing times were often delayed, which hindered the system's responsiveness during live analysis sessions. Furthermore, the ongoing need to retrain models with new data to maintain or enhance performance added to the computational burden, creating challenges for scalability and efficiency.

6.2.3 Scalability and Adaptability
As the system continued to develop and aimed to serve a wider range of golfers and playing styles, scalability emerged as a crucial concern. It was vital for the system to adapt to various golf courses, weather conditions, and equipment without necessitating extensive recalibration or retraining. This adaptability was essential for maintaining the system's relevance and usefulness over time, especially as golfers' techniques evolved and new trends emerged in the sport.

6.2.4 Time Constraints
A significant limitation was the lack of time available for the project. Coordinating schedules among team members, the company's supervisor, and the golf coach often created challenges in conducting timely data collection and analysis. This time pressure impacted the ability to explore all potential avenues for development, ultimately influencing the overall quality and comprehensiveness of the final system.

6.3 Future Work
In future iterations of this AI system, several key areas should be explored to enhance its functionality and effectiveness.

6.3.1 Expanding Dataset Collection
Improving the accuracy of the AI system relies on expanding dataset collection. Currently, there is a lack of availability and diversity in golf-specific datasets. Collaborating with more golf institutions and academies could provide access to a broader range of high-quality video footage and motion capture data, significantly enhancing the system's training capabilities.

6.3.2 Automated Golf Position Refining
Currently, the team manually breaks down video footage into individual frames and classifies each frame according to specific positions for training and testing purposes. Automating this process would significantly streamline the workflow, allowing for faster data preparation and reducing the potential for human error. The system could automatically identify and categorise frames based on golfer positions, facilitating a more efficient training process and enhancing the overall functionality of the analysis system

6.3.3 Frontend Tuning and integration of the backend
Next, we'll focus on further tuning the frontend elements. Client feedback will be collected to refine and ensure it continues to meet or exceed expectations. Furthermore, once the model has been deemed accurate by the team and has met the client's standards, the team will come together to implement backend functionalities. The team will then monitor the performance and accuracy if it still continues to function as per expectations.

7. CONCLUSION

The project's AI-driven golf swing analysis system implemented in this research has shown to be an effective tool for recognizing and evaluating critical swing locations, with an accuracy of 68%. To a large extent, the project's principal goal of developing a system capable of delivering real-time feedback on golf swings has been met. The system's great precision and memory in detecting certain poses demonstrate its potential to improve golf training approaches. However, the misclassification concerns reported in some classes underscore the importance of more data gathering and model refinement. Future studies should focus on increasing the dataset and improving the model's accuracy across all classes. This project establishes the framework for future advances in real-time swing analysis, offering useful input to golfers and coaches while also helping to the general enhancement of golf training approaches.

7.1 Summary of Key Points

The AI-driven golf swing analysis system demonstrated an overall accuracy rate of 68% in pinpointing crucial swing positions, showcasing high precision and recall in categories P1, P2, P3, P6, P8, and P9. However, misclassifications were prevalent in classes P4 and P10, necessitating additional scrutiny and data gathering. Despite these challenges, the project's primary achievement lies in creating a real-time golf swing analysis tool that delivers instant feedback to users, thereby revolutionising the training regimen for both golfers and coaches. This initiative represents a substantial leap forward in golf training methodologies, leveraging cutting-edge AI and computer vision technologies to offer a novel perspective on traditional coaching techniques and elevate player performance.

7.2 Lessons Learned

Throughout the project, the team grew significantly, both individually and collectively. A notable technical challenge faced was misclassification within the AI model. It is dealt with by extensively examining confusion matrices, pinpointing particular issues and strategizing on additional data collection methods. This process underscored the importance of data quality and the necessity for continuous refinement of the model. Additionally, this project offered valuable insights into machine learning and computer vision, enhancing the team's technical proficiency and deepening the understanding of these contemporary technologies. Furthermore, this initiative demonstrated the value of effective communication and teamwork, as collaboration was pivotal in overcoming barriers and achieving the project's goals. Looking ahead, future improvements will concentrate on expanding the dataset and optimising the model to enhance the system's accuracy and reliability. Overall, the project significantly enriched the team's comprehension of AI applications in sports and highlighted the potential for future innovation in this area.

REFERENCES

[1] Forgan G. Challenge the Culture of Coaching and Learn to Play Golf Again [Internet]. The Lost Art of Golf; 2024 [cited 2024 Jul 4]. Available from: https://www.thelostartofgolf.com/challenge-the-culture-of-coaching-and-learn-to-play-golf-again/

[2] He D, Liu R, Rehg J, Guo J. Visual Golf Club Tracking for Enhanced Swing Analysis [Internet]. 2012 [cited 2024 Jul 4]. Available from: https://www.researchgate.net/publication/240399424_Visual_Golf_Club_Tracking_for_Enhanced_Swing_Analysis

[3] McNally W, Akbarnejad A, Barbič J. GolfDB: A Video Database for Golf Swing Sequencing. In: Proceedings of the European Conference on Computer Vision (ECCV); 2020 Aug 23-28; Glasgow, UK [Internet]. 2020 [cited 2024 Jul 2]. Available from: https://paperswithcode.com/paper/golfdb-a-video-database-for-golf-swing

[4] Wang Y, Chen Z, Li H. Research on the Applications of Artificial Intelligence in Golf [Internet]. 2023 [cited 2024 Jul 2]. Available from: https://www.researchgate.net/publication/368491592_Research_on_the_Applications_of_Artificial_Intelligence_in_Golf

[5] Sharma S, Taneja P. A Survey on Golf Swing Analysis Using Computer Vision Techniques. In: Innovations in Computer Science and Engineering. Singapore: Springer; 2022. p. 337-348. (Lecture Notes in Networks and Systems; vol. 447). Available from: https://link.springer.com/chapter/10.1007/978-3-031-04881-4_28

[6] McNally W, Akbarnejad A, Barbič J. GolfDB: A Video Database for Golf Swing Sequencing. In: Proceedings of the European Conference on Computer Vision (ECCV); 2020 Aug 23-28; Glasgow, UK [Internet]. 2020 [cited 2024 Jul 4]. Available from: https://paperswithcode.com/paper/golfdb-a-video-database-for-golf-swing

[7] Lee J, Kim D, Park J, Lee S. AI-Based Golf Swing Analysis System Using Pose Estimation and Machine Learning Algorithms. Appl Sci (Basel). 2023;13(20):11227. Available from: https://www.mdpi.com/2076-3417/13/20/11227

[8] Chen Y, Zhu H, Zhang S. Vision-Based Golf Swing Analysis Using Deep Learning. IEEE Trans Neural Netw Learn Syst. 2020;31(8):2976-2984. Available from: https://ieeexplore.ieee.org/document/9157840

[9] Google Developers. Pose Landmarker [Internet]. Google; 2024 [cited 2024 Jul 26]. Available from: https://developers.google.com/mediapipe/solutions/vision/pose_landmarker

[10] TensorFlow. Pose Estimation with TensorFlow Lite [Internet]. 2024 [cited 2024 Jul 8]. Available from: https://www.tensorflow.org/lite/examples/pose_estimation/overview

[11] Microsoft. Confidence Score in QnA Maker [Internet]. 2024 [cited 2024 Jul 8]. Available from: https://learn.microsoft.com/en-us/azure/ai-services/qnamaker/concepts/confidence-score

[12] Wikipedia. Atan2 [Internet]. Wikipedia; 2024 [cited 2024 Jul 10]. Available from: https://en.wikipedia.org/wiki/Atan2

[13] The DIY Golfer. Swing Positions [Internet]. 2024 [cited 2024 Jul 20]. Available from: https://www.thediygolfer.com/swing-positions

GLOSSARY

Term	Definition
Artificial Intelligence (AI)	The simulation of human intelligence in machines, programmed for thinking and the emulation of human activities.
Computer Vision	It is an artificial intelligence field that trains computers to interpret and make decisions on visual data.
Deep Learning	It is a branch of machine learning that includes the use of a large number of deep neural networks that are artificial intelligence that can abstract a lot.
Neural Networks	A series of algorithms that behave like the neurons in the brain to reveal patterns from big data is a highly simplified explanation of Neural Networks.
Epoch	The number of epochs is a hyperparameter that defines the number of times that the learning algorithm will work through the entire training dataset.
MediaPipe Pose	An open-source cross-platform framework developed by Google for real-time pose estimation, tracking up to 33 landmarks of the human body.
TensorFlow Pose	A pose estimation framework from the TensorFlow machine learning library that employs deep learning techniques to identify 16 key body joints.
Data Preprocessing	The general process of getting raw data transformed in such a way that it will be applicable to perform a particular machine learning task. Typical steps include data collection, extraction, and augmentation.
Data Augmentation	The process of artificially creating new data from existing data in a bid to improve diversity within the training dataset and counteract class imbalance.
Pose Estimation	A computer vision task of estimating key body joints in images or video frames.
Edge Processing	The processing of data close to its source allows for in-network or real-time feedback and reduces time delays.
Confusion Matrix	A table is used to evaluate the performance of a classification model by comparing actual versus predicted classifications.
Landmark Detection	The process of identifying specific key points in an image that are used to understand the pose or structure of an object is commonly used in pose estimation.
Figma	A great UI/UX design tool is web-based one that is used to create interfaces, prototypes, and design systems.
Tailwind	A utility-first CSS framework is a very efficient instrument for the construction of particular designs without HTML.
FontAwesome	A powerful icon library is useful for adding style and visual presentation to the user interface.

APPENDICES

APPENDIX A: MEETING MINUTES

JEREMY LIM MIN EN
To: joo@starviewint.net
Cc: IVAN PHUA YOU WEN; LAI CHUNG SHING; KOH YI TONG; NICHOLAS PHOON KAI JIN; Purnima Murali Mohan; Peter Loh Kok Keong

Mon 5/13/2024 4:43 PM

Dear Joo,

I hope this email finds you well. Following our meeting earlier today, I wanted to provide you with a summary of our discussions for your review. Please find the minutes of the meeting below:

MINUTES OF MEETING #1
Date: 13/5/2024
Time: 1415 hrs

Purpose of stick figure: The stick figure is a simple representation of body posture in a 2D perspective. It can be used to measure initial setup, body position, spine tilt, arm bending, body movement during the golf swing, hand and wrist positions at various points in the swing, head movement, and other factors that a golf coach would look for. The stick figure will measure these aspects but will not be the guiding factor; AI models will guide instead.

Data displayed to the user: The data should be objective and scientific, showing small details like centimeters and small angles.

Detection of ball hit: Use the sound of the ball being hit as the detection signal. There should be limitations to eliminate background noise as much as possible.

Type of camera: Research into the best camera to use. Video quality and sound may be vital. Consider different framerates in the camera. It should not be too costly but must be good enough.

Research on architecture: Research the best architecture for the solution. Consider client-server model vs. local software. The client-server model is preferred if possible due to lower cost.

First checkpoint: Arrival of the training data after successfully drawing stickman, and collecting the dataset.

Meeting frequency: Bi-weekly meetings.

Please review the minutes and let us know if there are any corrections or additional points you would like to include. Additionally, if you have any questions or need further clarification on any of the discussed topics, please don't hesitate to reach out to the team.

Looking forward to your feedback.

Best regards,
Jeremy Lim
2203516
+65 96938344

Figure A1. Screenshot of Meeting Minutes 1

Follow-up on Meeting Discussion [SE_Team_13] on the 27th of May 2024

Dear Joo,

I hope this email finds you well. Following our meeting earlier today, I wanted to provide you with a summary of our discussions for your review. I have also attached the slides in the email. Please find the minutes of the meeting below:

MINUTES OF MEETING

Date: 27/5/2024
Time: 1330 hrs

All players may have different kind of swing and our aim is to try to train a "proper neutral swing"

2 type of views and the things to look out for:
Front view (Face on)
Managing the stability of the swing:

- Golfer head should be relatively steady by using spine to check
  - 90 degree on spine to make sure the top centered
- Keep the lower body still on top swing
- Lower body move with the target on downswing
- Shoulders and hips move in the direction of the swing on downswing.

Down the Line

- Spine should be stable but inclined to the ground.
- Tilted Spine in follow through should be maintained throughout
- Inexperienced golfers will have not enough tilt or a change in tilt after the swing follow through
- Swing must try to be within the planar V-shaped area to have good contact

Stack and tilt golf swing is the main technique to focus on/

Figure A2. Screenshot of Meeting Minutes 2

Video could be broken down into Checkpoints(P) where data points will be recorded.
These data points can be used to highlight the difference or incorrect movements which will be highlighted to the user. E.g. knee extension or hip movements.

Checkpoints mentioned
P1 Standing beside the ball,
P2 Shaft parallel to the ground
P3 Backswing
P4 Top of backswing
P5 Left arm parallel to the ground
P6 Shaft parallel on down swing
P7 Impact with the ball
P8 Shaft parallel to the ground
P9 Hands parallel to the ground
P10 Follow through

Future implementations to think about

- Different point of views of the golf swings like the Top or Back view
- Allow for playback
- Live camera feed to do live analysis
- Other accessories like a wrist device that provides haptic feedback on wrong positioning with live feedback

Launch monitor

- The Data from launch monitor can be very useful as it predicts the ball flight
- Golf ball flight data is more significant compared to the positioning of the body as there are many different ways to position

Dataset provided will be Front-on and Down the Line. No launch monitor at the moment (for the future)

Resources to look into:
GearsGolf
https://www.gearssports.com/golf-swing-biomechanics/#why-gears

OnForm Golf
https://www.onform.com/golf

Please review the minutes and let us know if there are any corrections or additional points you would like to include. Additionally, if you have any questions or need further clarification on any of the discussed topics, please don't hesitate to reach out to the team.

Figure A3. Screenshot of Meeting Minutes 3

Follow-up on Meeting Discussion [SE_Team_13]

Dear Joo,

I hope this email finds you well. Following our meeting earlier today, I wanted to provide you with a summary of our discussions for your review.
Here is the link to the Google drive for Rob to upload the swings: https://drive.google.com/drive/folders/1bwWlcqjy5Gc2A0AUOTnnH2u4_odX29YG?usp=sharing
Please find the minutes of the meeting below:

How do we differentiate the Ps that look similar:
Separation of Ps are differentiated based on the direction of the swing

Video
Video has to be higher Frames per second so that we can get more data set on the swing
Rob will record himself to get good quality training data so that our model can pick up and identify the different Ps

How many swings do we need:
Start with 5 swing

How we would want the video :

- Preferably the same swing
- Preferably get swings from same person from the 2 different angle
- Front view is tougher due to the space constraint to get the proper depth
- Don't have to hit the ball for each swing
- Driver and iron will have different swing styles so we will focus on just iron for now
- Swings can all be done in one video.

Video quality:

- As high FPS as possible (minimum 30)
- Landscape
- Slowmo if possible

Figure A4. Screenshot of Meeting Minutes 4

Frontend
Have the ability to see how different the uploaded swing is to an ideal swing
Color code the data points depending if they are in the range of the ideal data points

- Green - within the good range
- Yellow - just out of range
- Red - totally out of range

Datapoints will be the 4 positions of hip and shoulders
Coaches should be able toIdentify using the color and body point what to improve

What else should be seen on the website:
2 different access to the website

- Coach to see swings
- Students to upload their swing
  - See their past golf swing
  - Recommendation from golf coach

Results should be simplified and understandable

Future enhancement:
Train a generative AI model which when presented with a student swing, can identify and present a prescription to fix their issues

Things to look into:
Calibration
Distance has to be calibrated due to different scale of the videos

Please review the minutes and let us know if there are any corrections or additional points you would like to include. Additionally, if you have any questions or need further clarification on any of the discussed topics, please don't hesitate to reach out to the team.

Figure A5. Screenshot of Meeting Minutes 5

Follow-up on Meeting Discussion [SE_Team_13]

Dear Joo,
I hope this email finds you well. Following our meeting earlier today, I wanted to provide you with a summary of our discussions for your review. Here is the link to the meeting recording earlier:
https://drive.google.com/file/d/1UemTaVkwHoyJiQ-WD8gMuq6jir8ZMlmP/view?usp=sharing . I have also attached the research the team has done on the cameras we can use to aid the project. There are also some questions for Rob at the bottom to be reviewed at your earliest convenience.
Please find the minutes of the meeting below:

Changing to a new framework - MediaPipe

- More landmarks (features)
- From 17 to 33 key points

Challenges we faced

- Some positions may not be accurately classified
  - Use time sequencing to further classify
    - P1 will be at the start while P7 will be further down

Angles on features

- Getting as many relevant angles for the golfers swing
  - To classify if its a good swing
- **Check with Rob to get the areas to measure, eg. spine tilt or wrist angles**
- Once we get all the measurements, we can proceed to plot a statistical curve to record and compare variance
  - If fall within range, can classify as a good swing.
  - Color code whether it fall within range
- The measurement is different from type of swing
  - For Rob, its **Stack and Tilt**
    - May be different than other methodologies
    - **Don't mix different types of swings into training**

Frontend

- No issues for now
- To be finetuned

Figure A6. Screenshot of Meeting Minutes 6

Camera

- Is there any way to control the camera to start recording from the software?
  - To look into further
- Move from camera to raspberry pi with a camera which can interact with the software
  - **To be handled in the next academic year.**

Things to be done:

- Integrate Mediapipe
- Work with time sequence to aid classification
- Fine-tuning frontend

Questions to Rob:

- Can we get all the important data points to measure at each position, the importance of the datapoint at each position and a good angle for each data point at the position
  - Given our current level of knowledge, we would greatly appreciate it if you could provide us with the additional columns that are essential for this project.
- Here is the document for your input: https://docs.google.com/document/d/1IV3IoL-m9b8WLfCBrKwjDnovcx9iY4W3G2W6x4pZFno/edit?usp=sharing

Please review the minutes and let us know if there are any corrections or additional points you would like to include. Additionally, if you have any questions or need further clarification on any of the discussed topics, please don't hesitate to reach out to the team.

Figure A7. Screenshot of Meeting Minutes 7

APPENDIX B: CODE SNIPPETS

```
import os
import cv2
from tqdm import tqdm

def create_folder(folder_name):
    if not os.path.exists(folder_name):
        os.makedirs(folder_name)

def process_video(video_path, output_folder):
    video_name = os.path.splitext(os.path.basename(video_path))[0]
    video_output_folder = os.path.join(output_folder, video_name)
    create_folder(video_output_folder)

    vidcap = cv2.VideoCapture(video_path)
    total_frames = int(vidcap.get(cv2.CAP_PROP_FRAME_COUNT))
```

```
    print(f"Processing video: {video_name} ({total_frames} frames)")

    count = 0
    progress_bar = tqdm(total=total_frames, desc=f"Extracting frames from {video_name}", unit='frame')

    while True:
        success, image = vidcap.read()
        if not success:
            break
        frame_filename = f"{video_name}_frame{count}.jpg"
        cv2.imwrite(os.path.join(video_output_folder, frame_filename), image)
        count += 1
        progress_bar.update(1)

    progress_bar.close()
    vidcap.release()
    print(f"\n{count} images are extracted in {video_output_folder}.")

def process_all_videos_in_directory(directory_path, output_folder):
    create_folder(output_folder)
    for filename in os.listdir(directory_path):
        if filename.endswith(".mp4") or filename.endswith(".avi") or filename.endswith(".mov"):
            video_path = os.path.join(directory_path, filename)
            process_video(video_path, output_folder)

input_directory = 'videos/ToFrames'          # Directory containing video files
output_directory = 'output_frames'           # Directory to save extracted frames
process_all_videos_in_directory(input_directory, output_directory)

print("Converting video to frames is completed.")
```

Figure B1. Code Snippet for Converting Videos to Frames

```
import os, cv2, glob, argparse, math, numpy as np, pandas as pd, mediapipe as mp
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--dataset", type=str, required=True,
                help="path to dataset/dir")
ap.add_argument("-o", "--save", type=str, required=True,
                help="path to save csv file, eg: dir/data.csv")
args = vars(ap.parse_args())
path_data_dir = args["dataset"]
path_to_save = args["save"]
torso_size_multiplier = 2.5
n_landmarks = 33
n_dimensions = 3
landmark_names = [
    'nose', 'left_eye_inner', 'left_eye', 'left_eye_outer', 'right_eye_inner', 'right_eye', 'right_eye_outer', ' left_ear', 'right_ear', 'mouth_left',
'mouth_right', 'left_shoulder', 'right_shoulder', 'left_elbow', 'right_elbow', 'left_wrist', 'right_wrist', 'left_pinky_1', 'right_pinky_1',
 'left_index_1', 'right_index_1', 'left_thumb_2', 'right_thumb_2', 'left_hip', 'right_hip', 'left_knee', 'right_knee', 'left_ankle', 'right_ankle',
```

```
 'left_heel', 'right_heel',   'left_foot_index', 'right_foot_index', ]
mp_pose = mp.solutions.pose
pose = mp_pose.Pose()
class_list = os.listdir(path_data_dir)
class_list = sorted(class_list)
col_names = []
for i in range(n_landmarks):
    name = mp_pose.PoseLandmark(i).name
    name_x = name + '_X'
    name_y = name + '_Y'
    name_z = name + '_Z'
    name_v = name + '_V'
    col_names.append(name_x)
    col_names.append(name_y)
    col_names.append(name_z)
    col_names.append(name_v)
full_lm_list = []
target_list = []
for class_name in class_list:
    path_to_class = os.path.join(path_data_dir, class_name)
    img_list = glob.glob(path_to_class + '/*.jpg') + glob.glob(path_to_class + '/*.jpeg')  + glob.glob(path_to_class + '/*.png')
    img_list = sorted(img_list)
    # Read reach Images in the each classes
    for img in img_list:
        image = cv2.imread(img)
        h, w, c = image.shape
        if image is None:
            print(
                f'[ERROR] Error in reading {img} -- Skipping.....\n[INFO] Taking next Image')
            continue
        else:
            img_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            result = pose.process(img_rgb)
            if result.pose_landmarks:
                lm_list = []
                for landmarks in result.pose_landmarks.landmark:
                    # Preprocessing
                    max_distance = 0
                    lm_list.append(landmarks)
                center_x = (lm_list[landmark_names.index('right_hip')].x +  lm_list[landmark_names.index('left_hip')].x)*0.5
                center_y = (lm_list[landmark_names.index('right_hip')].y + lm_list[landmark_names.index('left_hip')].y)*0.5
                shoulders_x = (lm_list[landmark_names.index('right_shoulder')].x + lm_list[landmark_names.index('left_shoulder')].x)*0.5
                 shoulders_y = (lm_list[landmark_names.index('right_shoulder')].y +  lm_list[landmark_names.index('left_shoulder')].y)*0.5
                for lm in lm_list:
                    distance = math.sqrt(
                        (lm.x - center_x)**2 + (lm.y - center_y)**2)
                    if(distance > max_distance):
                        max_distance = distance
                torso_size = math.sqrt(
```

```
                (shoulders_x - center_x)**2 + (shoulders_y - center_y)**2)
            max_distance = max(
                torso_size*torso_size_multiplier, max_distance)
            pre_lm = list(np.array([[(landmark.x-center_x)/max_distance, (landmark.y-center_y)/max_distance,
                    landmark.z/max_distance, landmark.visibility] for landmark in lm_list]).flatten())
            full_lm_list.append(pre_lm)
            target_list.append(class_name)
        print(f'{os.path.split(img)[1]} Landmarks added Successfully')
    print(f'[INFO] {class_name} Successfully Completed')
print('[INFO] Landmarks from Dataset Successfully Completed')
data_x = pd.DataFrame(full_lm_list, columns=col_names)
data = data_x.assign(Pose_Class=target_list)
data.to_csv(path_to_save, encoding='utf-8', index=False)
print(f'[INFO] Successfully Saved Landmarks data into {path_to_save}')
```

Figure B2. Code Snippet for saving landmarks detected into a CSV file

```
# Load .csv Data
df = pd.read_csv(path_csv)
class_list = df['Pose_Class'].unique()
class_list = sorted(class_list)
class_number = len(class_list)
# Create training and validation splits
x = df.copy()
y = x.pop('Pose_Class')
y, _ = y.factorize()
x = x.astype('float64')
y = keras.utils.to_categorical(y)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
print('[INFO] Loaded csv Dataset')
model = Sequential([
    layers.Dense(512, activation='relu', input_shape=[x_train.shape[1]]), layers.Dense(256, activation='relu'), layers.Dense(class_number,
activation="softmax") ])
# Model Summary
print('Model Summary: ', model.summary())
model.compile( optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'] )
```

Figure B3. Code Snippet for Model Architecture

```
checkpoint_path = path_to_save
checkpoint = keras.callbacks.ModelCheckpoint(checkpoint_path, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
earlystopping = keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=20)
print('[INFO] Model Training Started ...')
# Start training
history = model.fit(x_train, y_train, epochs=200, batch_size=32, validation_data=(x_test, y_test), callbacks=[checkpoint, earlystopping])
print('[INFO] Model Training Completed')
print(f'[INFO] Model Successfully Saved in /{path_to_save}')
```

Fig B4. Code Snippet for Model Training

```python
# Plot History
metric_loss = history.history['loss']
metric_val_loss = history.history['val_loss']
metric_accuracy = history.history['accuracy']
metric_val_accuracy = history.history['val_accuracy']

# Construct a range object which will be used as x-axis (horizontal plane) of the graph.
epochs = range(len(metric_loss))

# Plot the Graph.
plt.plot(epochs, metric_loss, 'yellow', label=metric_loss)
plt.plot(epochs, metric_val_loss, 'red', label=metric_val_loss)
plt.plot(epochs, metric_accuracy, 'blue', label=metric_accuracy)
plt.plot(epochs, metric_val_accuracy, 'green', label=metric_val_accuracy)
```

Figure B5. Code Snippet for Plotting the Model Training's history

```python
# Confusion Matrix
y_pred = model.predict(x_test)
y_pred_classes = y_pred.argmax(axis=1)
y_true = y_test.argmax(axis=1)

cm = confusion_matrix(y_true, y_pred_classes)

plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_list, yticklabels=class_list)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')

confusion_matrix_png = os.path.exists('confusion_matrix.png')
if confusion_matrix_png:
    os.remove('confusion_matrix.png')
    plt.savefig('confusion_matrix.png', bbox_inches='tight')
else:
    plt.savefig('confusion_matrix.png', bbox_inches='tight')
print('[INFO] Successfully Saved confusion_matrix.png')
```

Figure B6. Code Snippet for Plotting Confusion Matrix

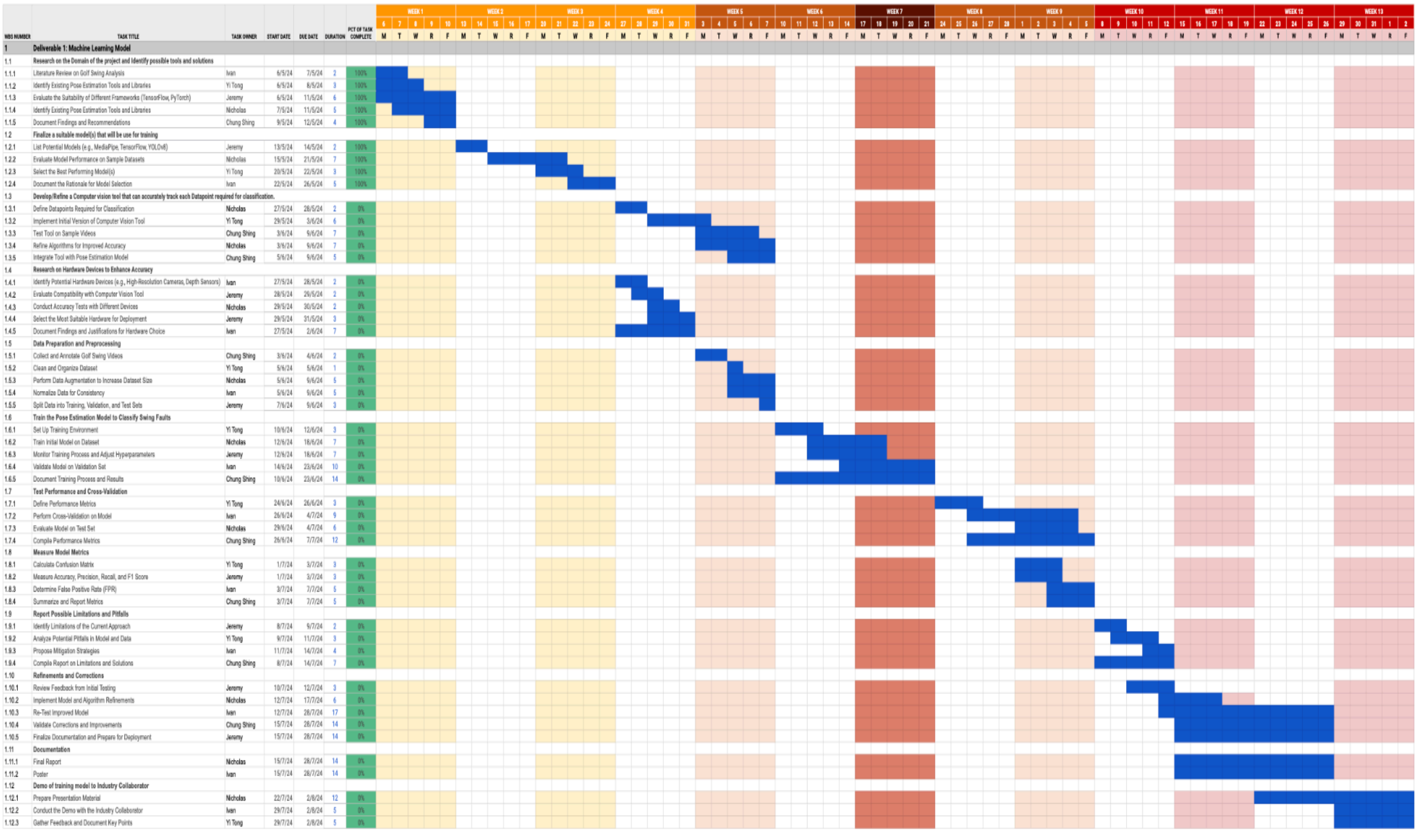
```python
ideal_shoulder_tilt = {
    'P1': 8, 'P2': 24, 'P3': 35, 'P4': 37, 'P5': 33,
    'P6': 12, 'P7': 30, 'P8': 38, 'P9': 45, 'P10': 6}

    # Calculate and print average shoulder tilt for each pose class
    good_postures = 0
    bad_postures = 0

    fig, axes = plt.subplots(5, 2, figsize=(15, 20))
    axes = axes.flatten()

    for i, (pose_class, angles) in enumerate(pose_class_angles.items()):
        if angles:
```

```
            ideal_angle = ideal_shoulder_tilt[pose_class]
            average_angle = sum(angles) / len(angles)

            # Calculate the standard deviation
            squared_diff = [(angle - ideal_angle) ** 2 for angle in angles]
            variance = sum(squared_diff) / len(angles)
            stddev = math.sqrt(variance)

            print(f"Standard deviation of shoulder tilt for {pose_class}: {stddev:.2f}
degrees")

            # Plot the normal distribution curve
            x = np.linspace(ideal_angle - 4*stddev, ideal_angle + 4*stddev, 100)
            y = stats.norm.pdf(x, ideal_angle, stddev)
            axes[i].plot(x, y,color='green', label=f'Normal Curve, std: {stddev:.2f}')

            # Fill the area under the curve
            axes[i].fill_between(x, y, color='green', alpha=0.2)

            # Plot the actual angles
            for angle in angles:
                angle = math.floor(angle)
                axes[i].axvline(x=angle, color='red', label=f'Predicted: {angle}')

            # Add line for ideal value
            axes[i].axvline(x=ideal_angle, color='blue', label=f'Ideal: {ideal_angle}')

            # Add lines for ±1 standard deviation from the ideal value
            axes[i].axvline(x=ideal_angle - stddev, color='orange', label=f'Ideal - 1
std: {ideal_angle - stddev:.2f}')
            axes[i].axvline(x=ideal_angle + stddev, color='orange', label=f'Ideal + 1
std: {ideal_angle + stddev:.2f}')
        else:
            axes[i].text(0.5, 0.5, f'No detections found for {pose_class}',
```

```
                    horizontalalignment='center', verticalalignment='center',
                    transform=axes[i].transAxes, fontsize=30, color='red')
    axes[i].legend()
    axes[i].set_title(f'Sample Prediction Outcome of {pose_class}')
    axes[i].set_xlabel('Shoulder Tilt Angle (degrees)')
    axes[i].set_ylabel('Probability Density')
    axes[i].grid()

plt.tight_layout()
```

Figure B7. Code Snippet for Plotting Sample Model Prediction Result

APPENDIX C: PROJECT MANAGEMENT



WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Deliverable 1: Machine Learning Model**					
1.1	**Research on the Domain of the project and Identify possible tools and solutions**					
1.1.1	Literature Review on Golf Swing Analysis	Ivan	6/5/24	7/5/24	2	100%
1.1.2	Identify Existing Pose Estimation Tools and Libraries	Yi Tong	6/5/24	8/5/24	3	100%
1.1.3	Evaluate the Suitability of Different Frameworks (TensorFlow, PyTorch)	Jeremy	6/5/24	11/5/24	6	100%
1.1.4	Identify Existing Pose Estimation Tools and Libraries	Nicholas	7/5/24	11/5/24	5	100%
1.1.5	Document Findings and Recommendations	Chung Shing	9/5/24	12/5/24	4	100%
1.2	**Finalize a suitable model(s) that will be use for training**					
1.2.1	List Potential Models (e.g., MediaPipe, TensorFlow, YOLOv8)	Jeremy	13/5/24	14/5/24	2	100%
1.2.2	Evaluate Model Performance on Sample Datasets	Nicholas	15/5/24	21/5/24	7	100%
1.2.3	Select the Best Performing Model(s)	Yi Tong	20/5/24	22/5/24	3	100%
1.2.4	Document the Rationale for Model Selection	Ivan	22/5/24	26/5/24	5	100%
1.3	**Develop/Refine a Computer vision tool that can accurately track each Datapoint required for classification.**					
1.3.1	Define Datapoints Required for Classification	Nicholas	27/5/24	28/5/24	2	0%
1.3.2	Implement Initial Version of Computer Vision Tool	Yi Tong	29/5/24	3/6/24	6	0%
1.3.3	Test Tool on Sample Videos	Chung Shing	3/6/24	9/6/24	7	0%
1.3.4	Refine Algorithms for Improved Accuracy	Nicholas	3/6/24	9/6/24	7	0%
1.3.5	Integrate Tool with Pose Estimation Model	Chung Shing	5/6/24	9/6/24	5	0%
1.4	**Research on Hardware Devices to Enhance Accuracy**					
1.4.1	Identify Potential Hardware Devices (e.g., High-Resolution Cameras, Depth Sensors)	Ivan	27/5/24	28/5/24	2	0%
1.4.2	Evaluate Compatibility with Computer Vision Tool	Jeremy	28/5/24	29/5/24	2	0%
1.4.3	Conduct Accuracy Tests with Different Devices	Nicholas	29/5/24	30/5/24	2	0%
1.4.4	Select the Most Suitable Hardware for Deployment	Jeremy	29/5/24	31/5/24	3	0%
1.4.5	Document Findings and Justifications for Hardware Choice	Ivan	27/5/24	2/6/24	7	0%
1.5	**Data Preparation and Preprocessing**					
1.5.1	Collect and Annotate Golf Swing Videos	Chung Shing	3/6/24	4/6/24	2	0%
1.5.2	Clean and Organize Dataset	Yi Tong	5/6/24	5/6/24	1	0%
1.5.3	Perform Data Augmentation to Increase Dataset Size	Nicholas	5/6/24	9/6/24	5	0%
1.5.4	Normalize Data for Consistency	Ivan	5/6/24	9/6/24	5	0%
1.5.5	Split Data into Training, Validation, and Test Sets	Jeremy	7/6/24	9/6/24	3	0%
1.6	**Train the Pose Estimation Model to Classify Swing Faults**					
1.6.1	Set Up Training Environment	Yi Tong	10/6/24	12/6/24	3	0%
1.6.2	Train Initial Model on Dataset	Nicholas	12/6/24	18/6/24	7	0%
1.6.3	Monitor Training Process and Adjust Hyperparameters	Jeremy	12/6/24	18/6/24	7	0%
1.6.4	Validate Model on Validation Set	Ivan	14/6/24	23/6/24	10	0%
1.6.5	Document Training Process and Results	Chung Shing	10/6/24	23/6/24	14	0%
1.7	**Test Performance and Cross-Validation**					
1.7.1	Define Performance Metrics	Yi Tong	24/6/24	26/6/24	3	0%
1.7.2	Perform Cross-Validation on Model	Ivan	26/6/24	4/7/24	9	0%
1.7.3	Evaluate Model on Test Set	Nicholas	29/6/24	4/7/24	6	0%
1.7.4	Compile Performance Metrics	Chung Shing	26/6/24	7/7/24	12	0%
1.8	**Measure Model Metrics**					
1.8.1	Calculate Confusion Matrix	Yi Tong	1/7/24	3/7/24	3	0%
1.8.2	Measure Accuracy, Precision, Recall, and F1 Score	Jeremy	1/7/24	3/7/24	3	0%
1.8.3	Determine False Positive Rate (FPR)	Ivan	3/7/24	7/7/24	5	0%
1.8.4	Summarize and Report Metrics	Chung Shing	3/7/24	7/7/24	5	0%
1.9	**Report Possible Limitations and Pitfalls**					
1.9.1	Identify Limitations of the Current Approach	Jeremy	8/7/24	9/7/24	2	0%
1.9.2	Analyze Potential Pitfalls in Model and Data	Yi Tong	9/7/24	11/7/24	3	0%
1.9.3	Propose Mitigation Strategies	Ivan	11/7/24	14/7/24	4	0%
1.9.4	Compile Report on Limitations and Solutions	Chung Shing	8/7/24	14/7/24	7	0%
1.10	**Refinements and Corrections**					
1.10.1	Review Feedback from Initial Testing	Jeremy	10/7/24	12/7/24	3	0%
1.10.2	Implement Model and Algorithm Refinements	Nicholas	12/7/24	17/7/24	6	0%
1.10.3	Re-Test Improved Model	Ivan	12/7/24	28/7/24	17	0%
1.10.4	Validate Corrections and Improvements	Chung Shing	15/7/24	28/7/24	14	0%
1.10.5	Finalize Documentation and Prepare for Deployment	Jeremy	15/7/24	28/7/24	14	0%
1.11	**Documentation**					
1.11.1	Final Report	Nicholas	15/7/24	28/7/24	14	0%
1.11.2	Poster	Ivan	15/7/24	28/7/24	14	0%
1.12	**Demo of training model to Industry Collaborator**					
1.12.1	Prepare Presentation Material	Nicholas	22/7/24	2/8/24	12	0%
1.12.2	Conduct the Demo with the Industry Collaborator	Ivan	29/7/24	2/8/24	5	0%
1.12.3	Gather Feedback and Document Key Points	Yi Tong	29/7/24	2/8/24	5	0%

Figure C1. ITP Estimated Time-line WBS