ELEC 490 Final Report:

Smart Squat Feedback Application

Group 52: Thierry Jones, Andrew Kroeger, Tayo Oduyemi, Sean Pollen

Presented to:

Dr. Alireza Bakhshai, P.Eng.

Dr. Mike Korenberg, P.Eng.

Dr. Alex Tait

Dr. Sean Whitehall

Dr Il-Min Kim, P.Eng.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

Background

Weightlifting, especially with heavy weights and/or complex movements is a high-risk sport. Not nearly enough people have the education to do the movements safely and traditional means of training, e.g., fitness trainers, buddy-system, self-learning are failing gym-goers. Many of these injuries can be attributed to poor form.

To address the lack of access to effective training, the team proposed The Smart Rack, an exercise rack that can observe a weightlifting movement, determine if the movement is done correctly, and if not, provide feedback to correct the movement. All done mid-workout set. Initially, the solution included a hardware system consisting of an exercise rack, image capturing devices, likely LiDAR or stereo cameras, an embedded computer, a Raspberry Pi 4 B, a Smart display device and iOS consumer app.

Solution

With further planning and ideation, the team elected to develop a completely software solution, broken down into two parts:

1. Exercise classification
2. Exercise scoring

To prove this concept, the team focused on the squat exercise and developed the Smart Squat Feedback Application. With this solution, a user records themselves on their laptops, performing a squat or another arbitrary movement. This video is fed to a MoViNet movement classifier model that uses a spatial/temporal (2+1)D convolution operation to extract features from frames. These features are then fed to multi-layer perceptron layers that then classify whether a squat exercise was performed or not. If a squat is detected, the system then moves to the squat scoring component. This component uses OpenCV to lay a skeleton over the user's body in the video and track the trajectories of joint keypoints on their body. This trajectory data and additional angular features generated therefrom are then fed into an algorithm that uses scores the quality of the squat based on different key criteria: squat depth, torso bend, foot angle, knee displacement.

Each of these criteria are scored out of five based on the kinematic features generated. The overall squat score is a weighted average of these values. The ranges and tiers of acceptable values were all taken from well-cited kinesiological studies. The capture of the video and the results can all be accessed through a PyQT desktop application.

Solution Evaluation

Unit-testing was employed for the to test both the squat detector and scorer components separately. Respectively, the most important success indicators were the accuracy of the movement classification and the accuracy of the feedback.

The squat detector was evaluated using a subset of the UCF 101 Dataset. This dataset contained videos of people conducting various movements such as push-ups, squats, and other exercises. To ensure the solution was robust, the dataset was sure to include different body types, athletic abilities, and variations of exercise. Various pre-trained variations of the MoViNet model were tested. Smart Squat accuracy of this component matched that of the pre-trained accuracy ranging from 71.5 – 82.7%.

To evaluate the squat scorer, the team measured the angles and displacements outputs from the skeleton overlay and compared that to real-life measurements. The team found the OpenCV model to be accurate. Additionally, in every case, the scores for the individual criterion matched the measurements taken from the skeleton overlay. In real-life integration testing, both the squat detector and evaluator were found to accurately detect and provide feedback for squat movements. The user interface was also evaluated with a focus group of five peers. All were able to successfully use the app without aid and the only feedback given concerned the colour scheme.

1.0 – Motivation and Background7y

1.1 - Motivation

1.11 - Weightlifting Related Injuries

Weightlifting is a widely practiced form of strength training among the millions of individuals worldwide who frequent gyms each year, seeking to improve their physical fitness. Though it is the most effective activity when it comes to building strength quickly, it is high risk. In fact, free-weight exercises are responsible for 52.6% of exercise related injuries, surpassing notoriously dangerous contact sports such as American football, ice hockey, and rugby [1]. 970,000 Americans endured weightlifting related injuries that justified an emergency room visit between 1990 and 2007 [2].The injury incidence rate in weightlifting is. 2.4-3.3 injuries per 1000 hours of training [3]. Many of these injuries can be attributed to poor form in four compound exercises including the squat, bench press, overhead press, and deadlift. Weightlifting has been proven to be a safe and effective training method when proper form is used. “Proper form” is an umbrella term describing a controlled lift consisting of proper posture and breathing while remaining within the scope of the athlete’s ability. On the contrary, improper form is the leading cause of weightlifting related injury.

1.12 - Personal Trainers & Inconsistent Information

The trivial solution to prevent these injuries is education. Athletes can either have an experienced friend accompany them to the gym, hire a personal trainer, or teach themselves proper lifting technique through the internet or other forms of media available to them. There is a myriad of issues that exist within these options. Personal trainers range from $40 to $70 per hour, which instantly prices them out of reach for less affluent athletes [4]. Additionally, there is no guarantee that the trainer they hire will help prevent injury. Canada does not have any laws regulating the standards of personal trainers [5]. The fitness industry is biased towards hiring trainers that “look buff” rather than those that have the necessary skills to educate their clients [6]. It is often up to the discretion of the client to decide whether their trainer is adequate, which for many is out of their ability to do. Unreliable access to quality exercise advice makes attending the gym with a friend or self-education insufficient in preventing injury. Fitness advice from social media platforms such as Instagram and TikTok often prioritize body aesthetics above safety and long-term health. Additionally, the never-ending stream of content from these services provides people with conflicting advice, often leaving them more confused than where they began.

1.13 - Inaccessible Existing Consumer Products

All existing products that help correct poor form for compound exercises fail to address accessibility obstructions. The flagship product in this segment is the Tempo Studio, an armoire shaped device that encourages users to accomplish as many repetitions as possible within a 60 second time frame for the selected workout. Real time feedback is generated using computer vision, sensors on weights, and artificial intelligence algorithms, which is later displayed on a 42-inch HD touch screen. The Tempo Studio has an MSRP of $2,495USD, and requires a $39 USD monthly subscription. The company offers a more affordable version listed at $495 USD, which integrates sensors on weights with the customer's iPhone camera [7]. Similar products such as Tonal (listed at $3,995 USD) also requires dedicated hardware and fails to help prevent workout injuries whilst adhering to accessibility constraints [8].

1.2 - Problem Statement

The Smart Squat Feedback System seeks to reduce the incidence of injury during compound lifts, with a core focus on accessibility. The system provides real time feedback to users completing the squat exercise and will soon provide feedback for the remaining three problematic compound lifts. The product requires no dedicated hardware beyond a laptop with a webcam, thus making it affordable and easy to use. In essence, the system observes two videos of a user performing an exercise. It then classifies the exercise performed (whether it be a squat, bench press, etc) and provides a feedback score based off body mechanics measured during the set. In its current iteration only scores for the squat are generated (hence the name SmartSquat). The application and its output are encapsulated in a streamlined user interface such that anybody can improve their form and be confident that they can exercise in absence of injury.

2.0 – Design

2.1 - Overview

The overall design objective of the application is to provide users with a simple interface that will allow them to record themselves performing a squat with their laptop and be provided with feedback on the quality of their form. Additionally, the application will use machine learning to recognize the type of exercise that's being performed by the user. The high-level flow diagram for the design can be seen below:

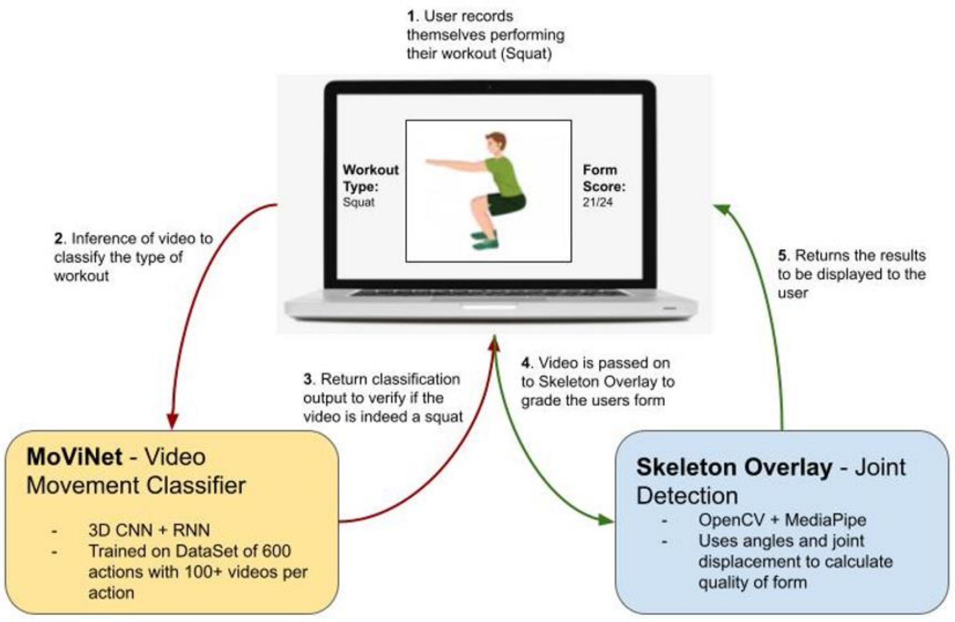

1. User records themselves performing their workout (Squat)
Workout Type: Squat
Form Score: 21/24
2. Inference of video to classify the type of workout
5. Returns the results to be displayed to the user
3. Return classification output to verify if the video is indeed a squat
4. Video is passed on to Skeleton Overlay to grade the users form
MoViNet - Video Movement Classifier
- 3D CNN + RNN
- Trained on DataSet of 600 actions with 100+ videos per action
Skeleton Overlay - Joint Detection
- OpenCV + MediaPipe
- Uses angles and joint displacement to calculate quality of form

Figure 1: High Level Design Flow Diagram

The three major components of the design were treated as their own subsystems throughout development.

2.11 - Computer Vision & Joint Detection

A form feedback script was developed to provide the analysis and grading of the user's quality of form. The script takes in an mp4 video file of a user performing a squat and then can create a skeleton overlay tracking all the relevant joints of the user (shoulders, knees, elbows, back, hips, etc.). Using the skeleton overlay angles and changes of the joint throughout the video can be used to create an algorithm that grades the quality of the form. This subsystem also required much research into what attributes contribute to a safe and effective squat.

2.12 - Exercise Recognition Machine Learning

The second subsystem was the exercise recognition neural network. The job of this component was to classify the type of exercise/action being performed by the user. This component would use an open-sourced action-recognition neural network called MoViNet [9]. There are two reasons for including the exercise recognition component. Firstly, the application can validate that a squat is indeed being performed and request a new recording if it detects that a squat is not being performed. Secondly, the long-term vision of this application is to have form feedback algorithms for a variety of exercises beyond just a squat and having this machine learning recognition enhances the user experience as they wouldn't need to manually enter which exercise, they're performing.

2.13 - User Interface

The last major subsystem is the user interface. This component is a python GUI that can easily be run on any laptop with a clean and simple design. It has a button allowing the user to record themselves with a 5 second countdown, and a button that will analyze the video providing feedback which gets nicely displayed in the GUI. There are also clear instructions on how to use the GUI when it is loaded up.

2.2 - Design Changes & Iterations

The final design has had several iterations since the beginning of the course. Changes to the approach have been made as superior methods and tools were discovered. These iterations are outlined in the list below:

1. The original design planned to calculate the squat form quality by developing and training a neural network. The solution would have the neural network input a video of a squat and output a score for each form criteria of the squat. The problem with this approach was that a massive dataset had to be prepared and manual classification of each video would have had to be

performed by a human. This was not feasible for the team to complete in time given the size of the dataset required. The team opted for a computer vision & joint detection feedback algorithm approach instead which was also the most common approach found in other similar projects.

2. The original design made during APSC 390 involved prototyping a physical device with several hardware components such as an RGB camera, Raspberry PI, Battery, and display. The team pivoted away from the prototype device and instead chose to develop the application on the existing hardware owned by the user (laptop, phone, etc.).
3. The original design also did not include the exercise recognition component. This component was added for the reasons mentioned in 2.12 - Exercise Recognition Machine Learning.
4. The exercise recognition component was first attempted as our own trained neural network, however due to dataset constraints the team pivoted to the pre-trained and open-sourced action-recognition neural network MoViNet.
5. The first iteration of the user interface was an attempt at a full stack web application. It used React, a front-end JavaScript framework for the front end, and the react-webcam package to download the user's squat video. The back end of the web application used Flask; a Python based server-side framework. The team quickly decided to shift to a purely python local application after encountering roadblocks regarding file conversions during week 8.

2.3 - Functional Requirements

SmartSquat should be able to detect a user's movements and provide real-time feedback on their technique, including advice on how to improve their form. To achieve this, a neural network for real-time object detection will construct a skeletal frame of the user, and an ML model will provide feedback that does not disturb the user. The interface will display performance scores and troubleshooting information on a touch screen display, which should be easy to use and aesthetically pleasing. The software should be preloaded onto the embedded computer and execute power up, with a simple menu to start a squatting session. SmartSquat should be able to analyze the user's performance within a range of 10-90 seconds in real-time. Additionally, hardware setup should be straightforward, with the embedded computer turning on with one push of a button, and the mount adjusted to fit the user's frame and height during the squat.

3.0 – Implementation

3.1 – Technical Implementation

3.11 Exercise Recognition – MoViNet

The exercise recognition component was implemented using an open-sourced and pre-trained action classifier neural network called "MoViNet" [9]. The neural network was trained on a the massive "Kinetics 600" dataset which has over 600 different actions with each individual action having between 500 to 1000 videos [10]. MoViNet uses a 3D CNN (2+1 D) architecture which includes two spatial and one temporal dimension across frames. MoViNet has six tiers that trade-off accuracy and computation time. These tiers are a0 to a5 where a0 is the least accurate but fastest, while a5 is the most accurate but slowest [9].

3.12 - Code Implementation

MoViNet was implemented using TensorFlow hub to load the model. The following "load_movinet_from_hub" was provided by the MoViNet developers. The largest portion of the MoViNet implementation involved data preprocessing. Firstly, videos had to be converted from mp4 to gif format. This was done using the "write_gif" function from the pip package "moviepy.editor" as seen in the Appendix under Machine Learning. The second step involved formatting the gif video into recommended input as seen in the Table.

Table 1 - MoViNet Recommended Input Formats

Model	**Recommended Input**
A0	172 x 172, 5 fps
A1	172 x 172, 5 fps
A2	224 x 224, 5 fps

A3	256 x 256, 12 fps
A4	290 x 290, 8 fps
A5	320 x 320, 12 fps

The A2 model was used in the implementation so the recommended input size is 224 x 224 @ 5fps. The recommended size was formatted in the “load_gif” function in Image Y. The recommended fps on the other hand was formatted using the pip package “moviepy.editor” and its function “set_fps” as seen in the Appendix.

3.2 – Exercise Evaluation – Skeleton Overlay

The second part of the smart squat application is the evaluation of the user’s form when performing the detected squat. The libraries used were OpenCV for joint detection from the MediaPipe video feed of the webcam the application is running on. OpenCV has Python bindings that allow developers to use the library in Python. The OpenCV Python module provides a variety of functions and classes to perform image processing tasks such as loading, displaying, and saving images, as well as implementing computer vision algorithms like object detection, tracking, and segmentation. Furthermore, the MediaPipe library provides a set of pre-built models for real-time media processing tasks, such as object detection, hand tracking, and facial recognition.

3.22 - Code Implementation

In terms of code implementation, OpenCV and MediaPipe libraries as well as numpy and time modules are imported, for efficient code implementation of the math used for the angle calculations.

Initially the video feed is captured by the webcam using OpenCV Figure 11- Calculate Angle Function, A MediaPipe Pose instance is then set up with the “min_detection_confidence” and “min_tracking_confidence” set to 0.5 or 50 percent. These parameters determine the minimum confidence level required for detecting landmarks and tracking over multiple frames. Detection happens in a while loop that runs while the video capture object is open, in this case a webcam, for testing the team decided 10 second recordings were ideal to record multiple repetitions of the squat. The frames are then recolored to RGB format using cv2.cvtColor(). Images are processed through the “pose.process()” function, which detects the landmarks in the image and returns the results for that frame. The results are then extracted from the “pose_landmarks” attribute of the frame object.

The landmarks of interest for the Smart Squat application and to calculate good form, include the left and right shoulders, hips, knees, and ankles, these are extracted from the results object using their respective PoseLandmark values. These landmarks are stored as lists of x and y coordinates, which can be used for angle and displacement calculations for further analysis. To calculate all these angles throughout the code the module numpy is used in a function that can be found in the Appendix Figure 11- Calculate Angle Function. The angles for the deepest leg bend and torso bend are stored in the variables “deepestLegBend” and “deepestTorsoBend”. The maximum vertical displacement of the shoulders and knees are stored in “max_shoulder_disp” and “max_knee_disp”. A range of angles based on studies outlined later is set up for squat form based on the values of “deepestLegBend”, “deepestTorsoBend”, “max_shoulder_disp”, and “max_knee_disp”. The range of angles is assigned a rating, ranging from 4-1, green, orange, yellow and red. The rating is based on how closely the angles match the ideal range for the squat form. For example, there is a code snippet in the Appendix Figure 13 - Squat Depth Ranges/Ratings that shows the ranges of angles for the squat depth calculated after the video has been processed. The ratings and overall scores are printed to the console and sent into the Qt user interface.

3.23 – Research and Backing Studies

The team conducted extensive research to determine the optimal ranges for proper squat form, relying on studies that identified the most effective positions for the joints involved in the exercise. To achieve optimal form, it was found that the shoulders must be kept in balance, and the knees should not move inward during the squat's depth. The team also considered other factors when determining optimal squat form, which are discussed below.

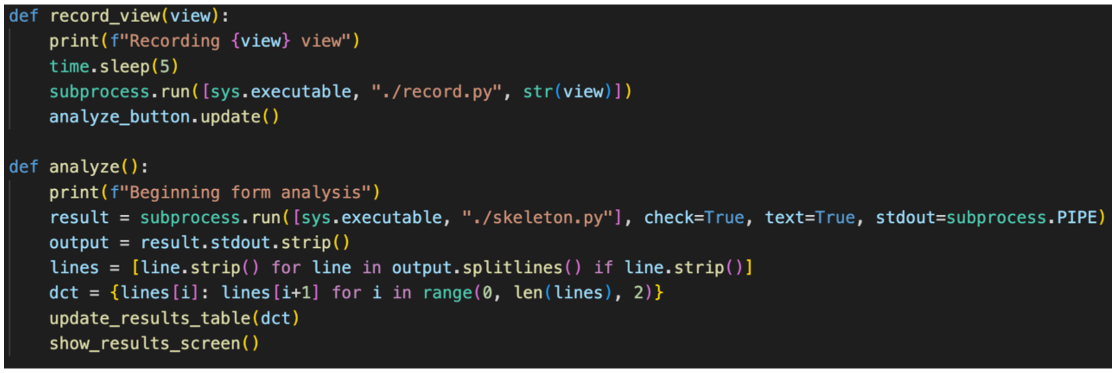
1. Squat Depth, according to the study “Difference in the magnitude of muscle damage between maximal and submaximal eccentric loading”, [11] a squat depth of approximately 90 degrees of knee flexion or parallel to the ground. This was found to result in maximal activation of the quadriceps muscle group. This depth is often considered optimal because it allows for maximal muscle activation while also minimizing the risk of injury. Going deeper than this can increase the risk of knee injury, while not going deep enough can limit the benefits of the squat.
2. Torso Bend, the study "Optimizing Squat Technique" concluded that excessive forward trunk lean beyond 50 degrees may increase the risk of spinal injuries, while keeping the torso too upright, less than 30 degrees, would reduce the activation of the hip extensors and gluteal muscles, leading to reduced performance and potential knee injuries. [12]
3. Foot Angle, "How to squat? Effects of various stance widths, foot placement angles and level of experience on knee, hip and trunk motion and loading" examined the effects of different foot placements on muscle activation and kinematics during the barbell back squat [13]. The researchers found that a foot placement angle of 42 degrees also resulted in a more vertical shin angle, which reduced stress on the knee joint and made the squat movement safer. The study suggests this foot angle helps individuals maximize muscle activation and joint loading during the barbell back squat while minimizing injury.
4. Knee Displacement: The study above also concluded that proper knee alignment during the squat is critical for minimizing the risk of injury to the knee joint. Specifically, the knees should be aligned with the toes throughout the movement, and excessive inward movement should be avoided.

Thus, optimal ranges for all these characteristics were set to reflect the research above. All these can be seen in the Appendix in the code snippets for the skeleton overlay.

3.3 – User Interface

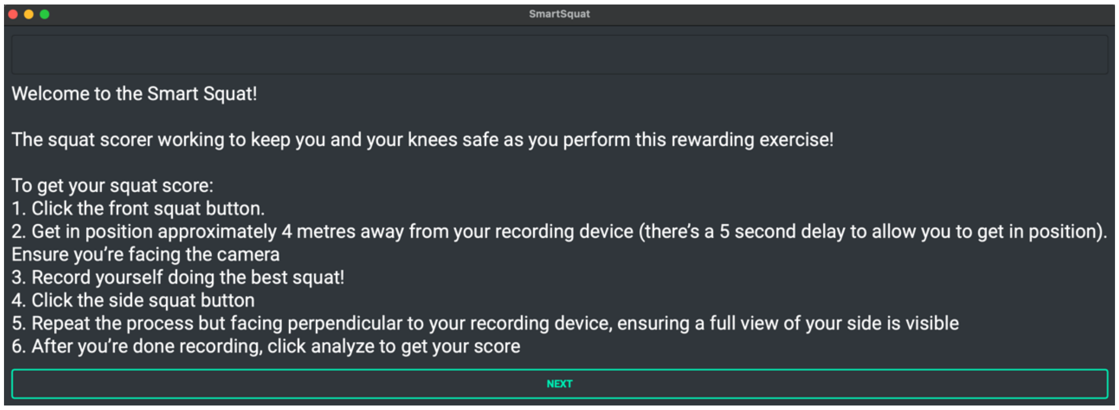
The user interface consists of 5 screens developed using Python and PyQt5 modules. The first is an instruction screen that explains the purpose of the application and how it works. It is designed such that a user without any information about the application can figure out how to operate it. This was by design, as the target demographic may not have the technical expertise to operate a confusing application. After reading the instruction screen and clicking "next" the user is lead to the main menu. Here, they can record front and side views of their squats as shown in Figure 5 - UI Skeleton Overlay Playback (Side View). Once front and side views are recorded the user can click the "analyze form" button, where their videos are played back to them with the skeleton overlay attached. Once the videos are finished being played, the scores and measurements are generated to populate the table in Figure 7 - UI Feedback Table. After clicking "done" the user is sent back to the main menu, they can re-watch their recordings or record a new set.

Each button is organized in a PyQt widget with a vertical layout. When "next" and "back" buttons are clicked the appropriate buttons or features are hidden or shown corresponding to each screen. For action buttons such as "record front view", "record side view" and "analyze form" a python subprocess (in other words, a different script) is called when clicked. Videos are stored locally in the project repository with the title "front.mp4" or "side.mp4" respectively. Only one file with each of these names can exist in the directory. If a new video is recorded the previous one is overwritten. These videos being stored locally allows the application to be run without an internet connection. To populate the feedback table, text from the skeleton overlay output is parsed and stored in a dictionary.



```python
def record_view(view):
    print(f"Recording {view} view")
    time.sleep(5)
    subprocess.run([sys.executable, "./record.py", str(view)])
    analyze_button.update()

def analyze():
    print(f"Beginning form analysis")
    result = subprocess.run([sys.executable, "./skeleton.py"], check=True, text=True, stdout=subprocess.PIPE)
    output = result.stdout.strip()
    lines = [line.strip() for line in output.splitlines() if line.strip()]
    dct = {lines[i]: lines[i+1] for i in range(0, len(lines), 2)}
    update_results_table(dct)
    show_results_screen()
```

Figure 2 - Code snipped for 'record __ view' and 'analyze form' buttons



Welcome to the Smart Squat!

The squat scorer working to keep you and your knees safe as you perform this rewarding exercise!

To get your squat score:
1. Click the front squat button.
2. Get in position approximately 4 metres away from your recording device (there's a 5 second delay to allow you to get in position). Ensure you're facing the camera
3. Record yourself doing the best squat!
4. Click the side squat button
5. Repeat the process but facing perpendicular to your recording device, ensuring a full view of your side is visible
6. After you're done recording, click analyze to get your score

Figure 3 - UI Instructions Screens

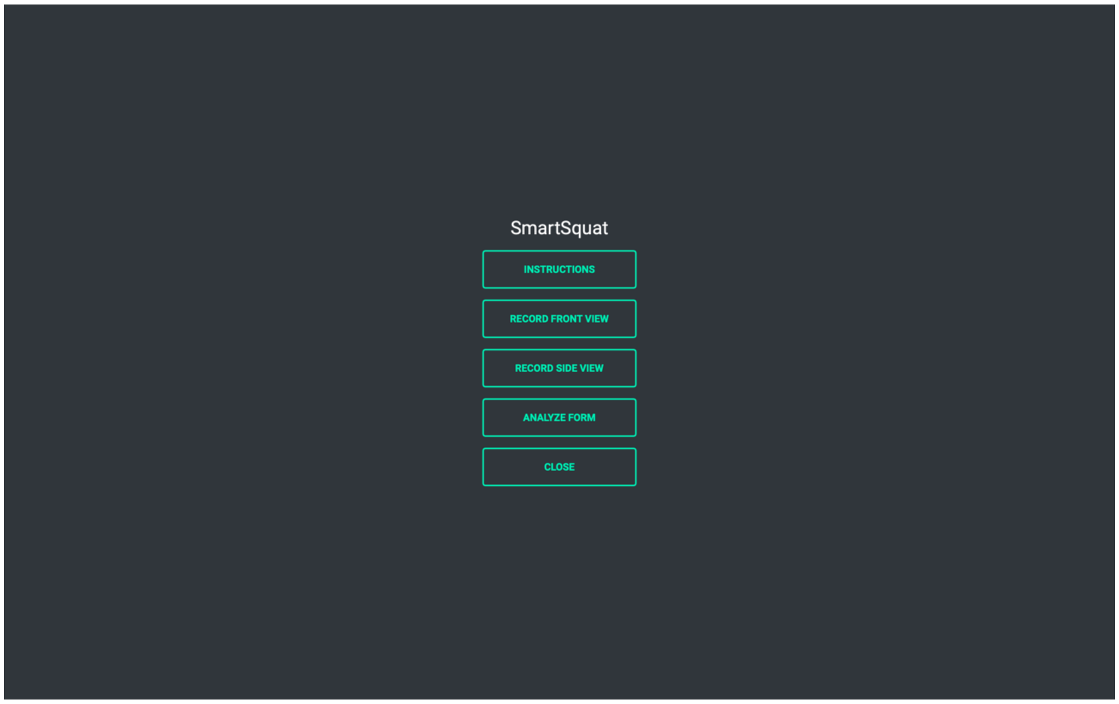


Figure 4 - UI Home Screen

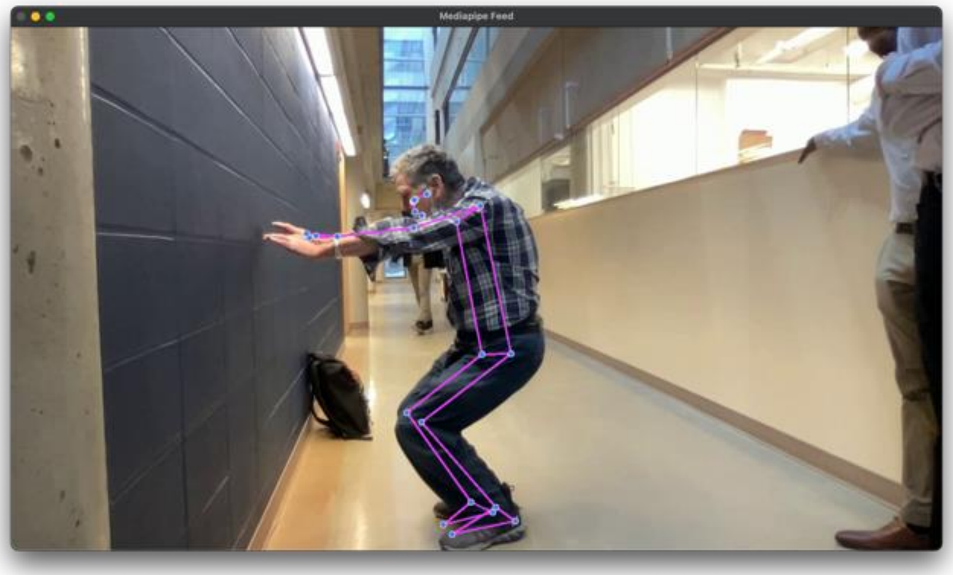


Figure 5 - UI Skeleton Overlay Playback (Side View)

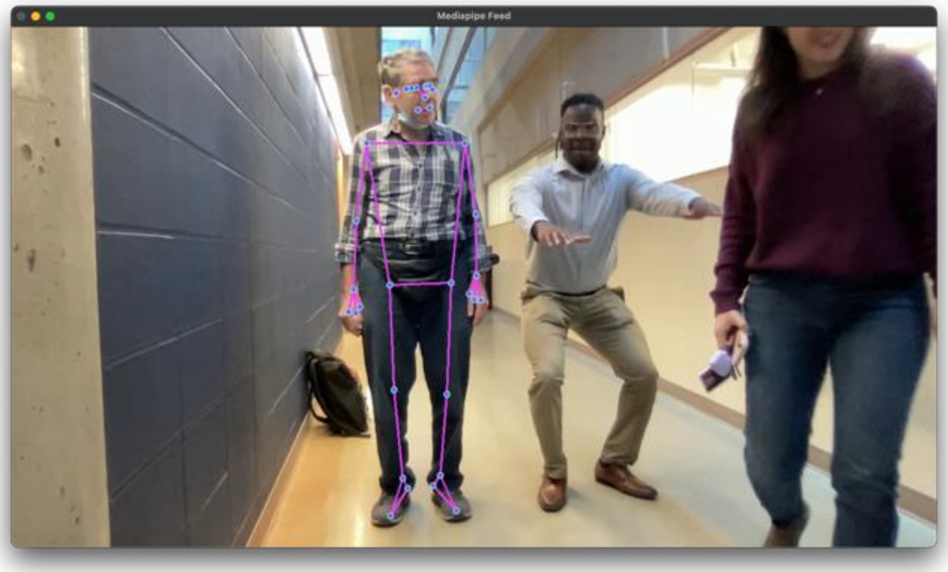
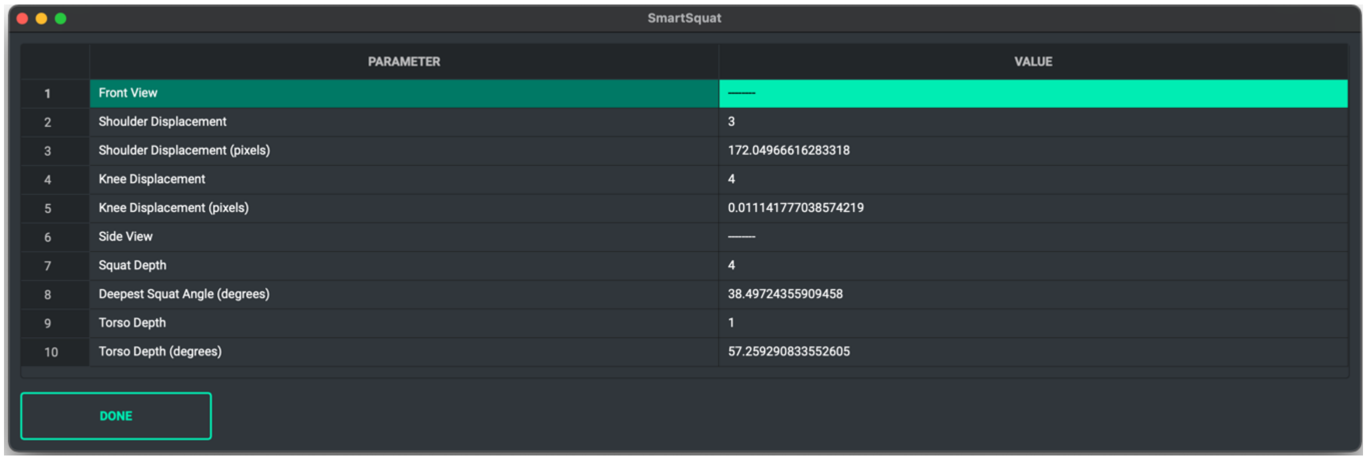


Figure 6 - UI Skeleton Overlay Playback (Front View)



	PARAMETER	VALUE
1	Front View	------
2	Shoulder Displacement	3
3	Shoulder Displacement (pixels)	172.04966616283318
4	Knee Displacement	4
5	Knee Displacement (pixels)	0.011141777038574219
6	Side View	------
7	Squat Depth	4
8	Deepest Squat Angle (degrees)	38.49724355909458
9	Torso Depth	1
10	Torso Depth (degrees)	57.259290833552605

Figure 7 - UI Feedback Table

3.4 – Budget

The budget in the blueprint projected a total expenditure of $540. After pivoting the project away from dedicated hardware, nothing needed to be purchased. The only material used for the project was a laptop with an integrated webcam already owned by the group. The software can run on most machines that university students and faculty would own. Therefore, the final cost of the project was $0, well within the initial budget.

Table - Bill of Materials

Component	Description	Supplier	Units	Estimated Cost
Laptop with integrated webcam.	2020 MacBook Air was used during the demo.	Group 52	1	$0

3.5 Adapted Timeline

Table 2 - Adapted Timeline

No.	Milestone	Due date	Date Completed	Member(s) Assigned
1	Design Phase 1. Order components 2. Finalize software architecture 3. Select external datasets	Week 12	1,2,3. January 7th (Start of week 13)	Thierry Tayo Sean
2	Development Phase 1. Test hardware components 2. Set up development environments, git repository, and JIRA. 3. Write code (computer vision component) 4. Write code (deep learning component)	Week 13	1.No hardware components ordered. 2. January 10th 3. January 30th 4. February 16th 5. Did not need testing locations since the team used a laptop webcam	Thierry Sean Sean + Thierry Andrew + Tayo Thierry

	5. Contacted relevant faculty members for access to testing location.			
3	Integration Phase / Testing Phase 1. Complete minimum viable product. 2. Begin user testing on subjects in real time. 3. Add internal dataset if required	Week 19	1. March 6th 2. March 10th 3. Did not enhance the data set since the team used MoViNet to detect the squat.	Entire Team Entire Team Andrew + Tayo Sean + Thierry
4	Complete Testing Phase 1. Implement final tweaks to model. 2. Implement final tweaks to image recognition.	Week 20	1.March 16th 2.March 20th	Tayo + Andrew Sean + Thierry
5	Additional time to fix outstanding issues Open House	Week 22	March 23rd	All members
6	Final deliverable	Week 23	March 30th	All members
7	Final project report, presentation	Week 24	April 3rd	All members

4.0 - Testing & Evaluation

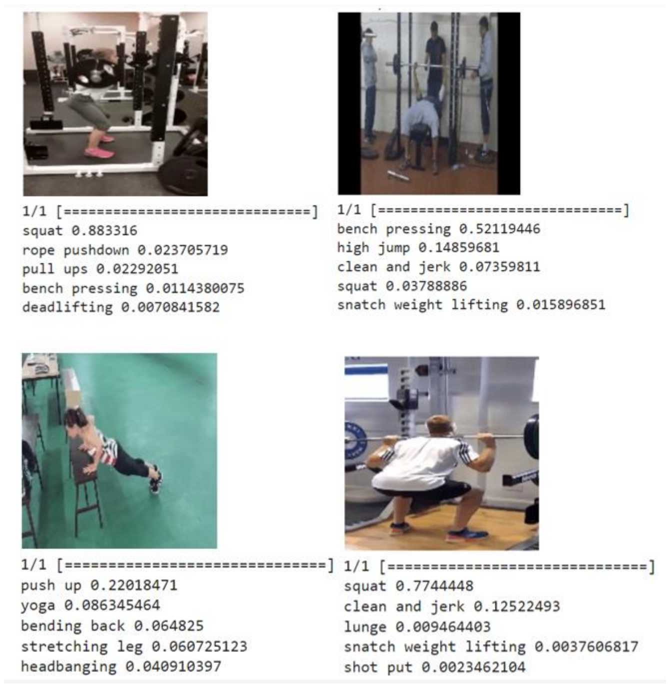
4.1 - Exercise Recognition Machine Learning

Since the MoViNet neural network architecture is pre-trained, the training & testing accuracies are publicly available. In addition to the public data the team ran custom evaluation for accuracy using a separate dataset.

4.11 – Accuracy and Performance Evaluation

The performed testing of the MoViNet accuracy using the UCF 101 Dataset. This dataset provided a large list of YouTube links categorized into 101 different workout types [14]. The team used a python script to download and format each video down to a 10 second clip of the video. Using a small subset of the dataset the team tested each tier (A0 – A5) of MoViNet for its accuracy and computation time.

The evaluation data-subset used contained 100 videos with a mix of squats, pushups, and bench-press actions. A prediction was considered correct if the action that received the highest-weighted probability by the model was the indeed the actual accuracy (otherwise known as Top-1 accuracy). Here are some examples of correct classifications:



```
1/1 [==============================]
squat 0.883316
rope pushdown 0.023705719
pull ups 0.02292051
bench pressing 0.0114380075
deadlifting 0.0070841582
```

```
1/1 [==============================]
bench pressing 0.52119446
high jump 0.14859681
clean and jerk 0.07359811
squat 0.03788886
snatch weight lifting 0.015896851
```

```
1/1 [==============================]
push up 0.22018471
yoga 0.086345464
bending back 0.064825
stretching leg 0.060725123
headbanging 0.040910397
```

```
1/1 [==============================]
squat 0.7744448
clean and jerk 0.12522493
lunge 0.009464403
snatch weight lifting 0.0037606817
shot put 0.0023462104
```

Figure 8 - Exercise Classification Examples

Each example shows the descending order of the top 5 highest weighted actions predicted by the model.

These images show unique examples of the different exercises being conducted with different body types, camera angles, and variations. In general, the more obvious the image the higher the prediction confidence. For example, the video on the bottom right showed a girl performing inclined pushups. This variation from the traditional exercise caused the confidence in the prediction to be much lower at only 22%.

The table below shows the results of the teams testing on the 100-video data subset. The hardware used during testing was a modern laptop with an AMD Ryzen 5 4500U (2.38 GHz) CPU. This is a very average hardware configuration, and many users of the application would have something similar.

Table 3 - MoViNet Top-1 Accuracy and Computation Time

Model	**Top-1 Accuracy (%)**	**Computation Time Per Video (seconds)**

A0	75	32
A1	78	37
A2	83	40
A3	85	48
A4	86	53
A5	88	62

The team's testing accuracy was comparable to the publicly available accuracy of MoViNet which is summarized in the table below.

Table 4 - MoViNet Publicly available Top-1 Accuracy

Model	**Top-1 Accuracy (%)**
A0	71.5
A1	76.0
A2	77.5
A3	80.8
A4	81.2
A5	82.7

4.12: Comparison to other Models

MoViNet performs very well when compared to other action-classification neural networks. The Graph below was taken from the MoViNet research paper and shows the comparison of three different neural networks. MoViNet (Base & Stream), X3D (10-Clip & Single-Clip), and MobileNet [9].

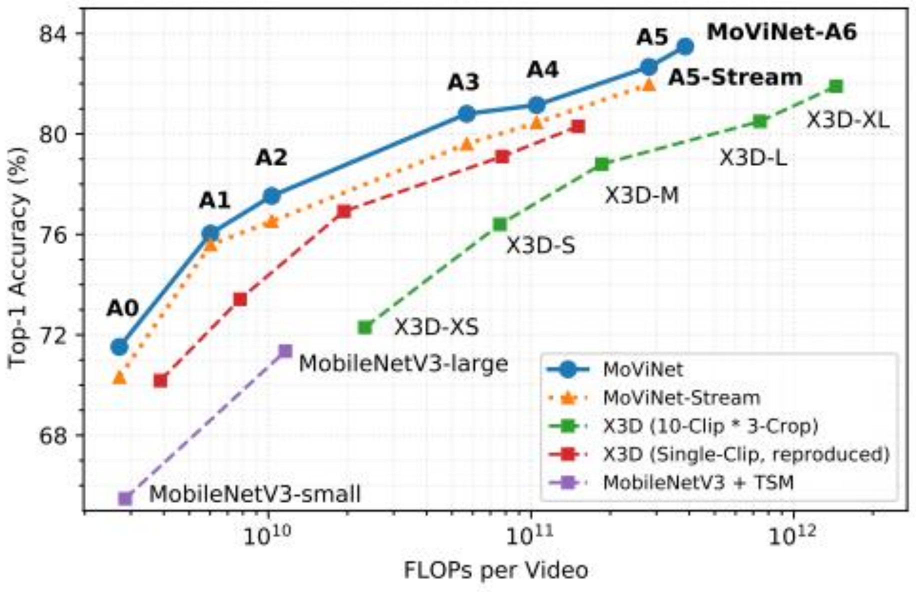


Figure 9 - Action-Classifier Comparison Graph

The graph shows that MoViNet outperforms the other models in Top-1 Accuracy per FLOP making it overall a more efficient action-classifier.

4.2 - Exercise Evaluation – Skeleton Overlay (Thierry)

The skeleton overlay underwent 3 types of testing. First was usability testing, to do this the team gathered a group of fitness-oriented students. They were asked to perform squats while recording themselves on their laptop. The system was evaluated for ease of use, accuracy of feedback, and overall satisfaction. The users were also asked to provide feedback on any issues they encountered or improvements that could be made to enhance the usability of the Smart Squat Application. Adjustments were made to the software interface and feedback mechanisms, as well as scoring ranges. Video recordings and data such as this video below capturing a frame from the recording.

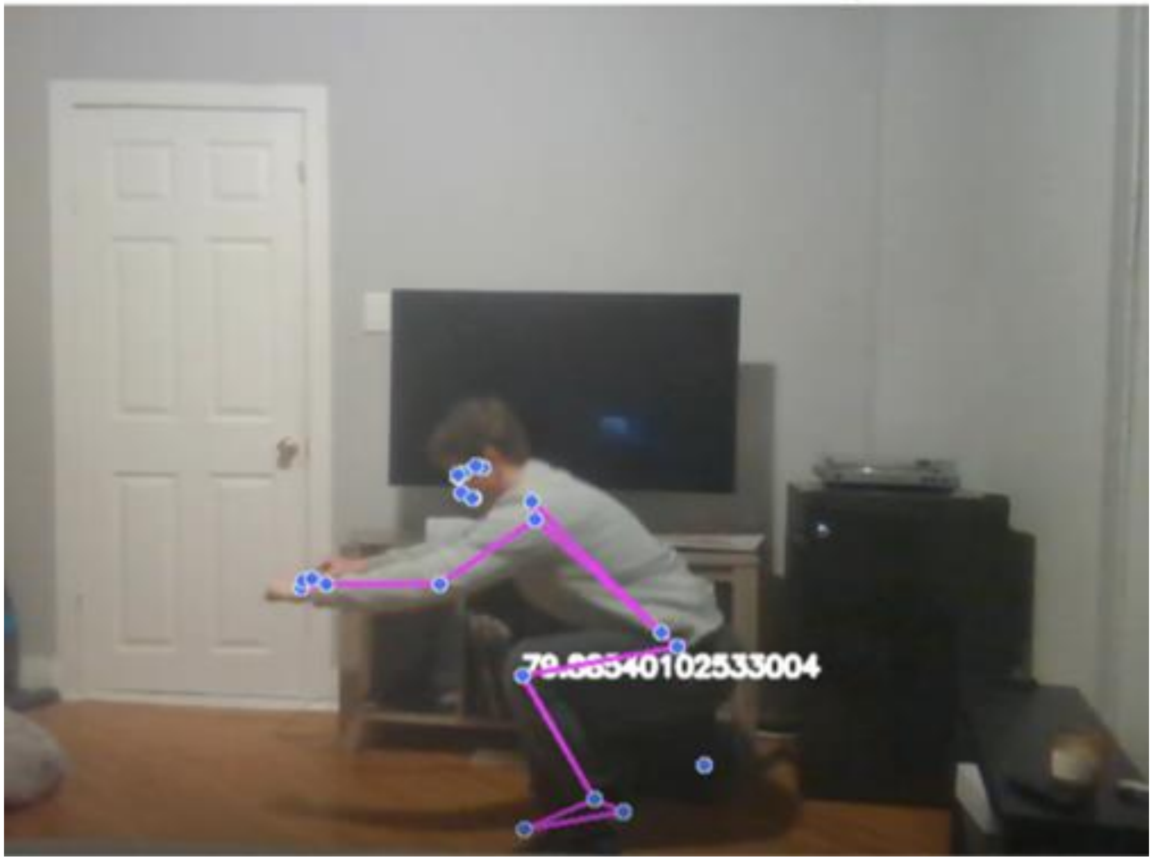


Figure 10 - Skeleton Overlay Testing

The second type of testing that was conducted was accuracy testing. This involved comparing the angles measured by the Smart Squat form corrector to those seen on the video recording to determine any errors in the joint detection. This was affected by positioning, lighting, and distance away from the camera. This allowed for the assessment of the application’s ability to accurately track joints, calculate angles, and provide feedback on squat form. The results of this testing were used to refine the ranges used in the algorithm to ensure that it was able to measure accurately and consistently how good or bad the squat recorded was.

The third type of testing was performance based. This was used to assess the application’s ability to perform in a variety of conditions and environments. For example, the team tested the Smart Squat in low-light conditions, dimming the lights to see how well the joint detection could do in a darker setting. Additionally, the team tested the skeleton overlay with all kinds of different users, with varying heights, body types, both men and women. By testing the system in a variety of different conditions, it was possible to ensure that it was robust and reliable in a range of real-world scenarios.

Furthermore, there are many approaches comparable to the team’s use of OpenCV joint detection and angle calculations to determine form grading. For instance, "Real-Time Multi-Person 2D Pose Estimation using Part Affinity Fields" presented a system that uses a convolutional neural network to detect joints on the body and construct a skeletal frame to estimate the pose of the human body [15]. This is comparable to the Smart Squat however it uses a different approach to attain joint detection.

4.3 – User Interface Testing

The user interface was briefly tested the week of the open house showcase. A group of 5 peers were selected that varied in their interest in fitness as well as their self-proclaimed technical ability. This was done to ensure that feedback regarding the interface was appropriate for a wide range of users. Users were not given any briefing regarding what the application is for, and they were told that they could not ask any questions during their trial. A team member observed how they used the application and took note of any errors if any occurred. Though the sample size was small, all users were effectively able to record their own squats without any assistance, deeming the user interface a success. After the trial when users were asked for feedback 3 of them had no comments, but 2 said to change the colour scheme to make the application seem more “polished”. A PyQt5 stylesheet was applied to make button accents green. The original style is shown in Figure 17 - Original UI Colour Scheme.

5.0 – Social, Safety, Environmental, Economic, Regulatory Compliance and Professional Considerations

The Smart Rack has much potential to improve the standard of living and health of the general population. Making the guidance of a fitness trainer accessible to all gym-goers will have the likely effect of reducing injury rates and maximizing the performance and results gained from The Smart Rack. These benefits come with concerns that must be addressed to minimize risk to the individual and society.

5.1 - Safety/Health Considerations

The Squat Feedback Application, offering feedback to weightlifters that may be under heavy load risks causing chronic or acute injury if the suggestion is inaccurate or unclear. Additionally, the algorithm used to score squat form is based on medical consensus on what *generally* constitutes a good squat. Every potential user has different fitness, mobility and health capabilities that must be considered when performing any exercise. These factors are not considered in the algorithm used. To mitigate any risk of inaccurate squat feedback due to individual fitness/health factors, it must be stressed that this solution is only to be used as a suggestive device, and their awareness of their own body, counsel from doctors and physiotherapists must always take precedence over the advice given by the squat feedback solution. Further, future iterations of this solutions should account for these factors through individual user questionnaire to adjust the feedback given. As the feedback accounts for more factors, all care must be made to ensure that the suggestions are based on medical and kinesiological consensus.

The feedback given by the solution must also be clear, easily interpreted and delivered in a fashion that allows the user to process the feedback, calmly make an adjustment, and continue their workout. Meaning multiple visual/auditory cues should be given. Currently, the proof of concept consists of several scores out of four for several attributes of the squat and the measurement from which that score is derived. Future work would include enhancing this feedback to be clearer, by letting the user know exactly what needs to be done to correct the squat form.

5.2 - Social Considerations:

There is a great disparity in societal health and fitness between developed and developing nations. Though this is caused by many environmental, social, and economic factors. This inequality is further compounded by access to community fitness centers, sports clubs, and gyms. A report from the *Global Wellness Institute* found that approximately 29.1% of the North American population participated in fitness at fitness clubs as opposed to 0.7% in Africa, 1.1% in the Asia-Pacific region and 1.4% in the Middle Eastern region [9]. In these developing countries, access to existing fitness facilities is overwhelmingly skewed towards the upper-class whereas in North America and Europe gyms are more accessible to middle-class populations. Further, in developed countries, access to fitness facilities is again skewed to those in higher income brackets. In fact, Americans making $100,000 or more annually, make up 38.98% of gym goers with gym membership plummeting when looking at people earning less than $75,000 annually [16]. This creates an innate disparity in fitness. Of course, one can lead a healthy lifestyle without access to gym and fitness equipment. Though, to match the results of a person with a gym, one would have to dedicate additional time and resources that they may not have. The Smart Squat Feedback Application, being introduced in North America, the biggest fitness market on Earth, would only serve to increase the gap between fitness in North America and other parts of the world. To mitigate this gap The Smart Squat team has made the system as accessible as possible. Due to it being a mostly software solution, the system can be used in any setting whether it be at the gym, at home or at a park. This mobility will be further compounded with the addition of a mobile app, which could be a future iteration of the solution. This is stark contrast from the initial solution of creating a smart exercise system. By limiting the hardware in the solution, the team has made fitness advice more accessible. Though a revenue generating scheme would need to be formulated, the cost of using our application almost certainly be cheaper than both a gym membership and/or personal training.

5.3 - Economic Risks:

As can be seen in any job market in which automation is involved, the team is anticipating a decline in the job growth for personal trainers, if The Smart Squat system becomes widespread in use. According to the

US Bureau of Labor Statistics, the job market for fitness trainers is expected to grow 15% from 2019 to 2029, due to more organizations recognizing the benefits of health and fitness programs for their employees and an overall culture shift towards healthy living and fitness [17]. This report was conducted without considering the impact smart technology and the introduction of internet of things technology to the fitness equipment sector. Up until quite recently the industry standard has been that gym equipment consists of metal, concrete, and rope among other elementary materials. Therefore, there is plenty of opportunity for smart technology to upend fitness equipment field and the fitness experience. As can be extrapolated from other sectors that have been introduced to automation, namely transportation, and manufacturing, this introduction will likely result in the decline of the use of traditional human personal trainers. Unfortunately, there is no easy way to mitigate this risk. Society must resolve the question of how we are going to introduce innovation and automation and balance that with the human workforce.

5.4 - Environmental

This solution is an entirely software solution. Thus, at its current iteration it does not have any environmental impacts beyond those caused by standard phone and laptop use. If the solution was to go to market and additional data centers and servers were needed to host the service, the negative ecological impacts of the solution begin to appear. The solution would likely be deployed via a cloud service such as Amazon Web Services (AWS) or Microsoft Azure. Though using a cloud service would have a smaller carbon footprint than operating a standalone datacenter for the solution, cloud services nevertheless have a massive carbon footprint. Indeed, a single cloud datacenter can use the equivalent electricity of 50,000 homes to power and cool the server. According to AWS, this is still about 88% less than the scaled consumption of operating a single service datacenter [18]. Thus, it is still worth hosting the solution in the cloud.

Beyond the carbon footprint associated with compute power, there are no adverse ecological impacts associated with the Smart Squat. This is due to the complete elimination of the hardware component of the initial design and a complete refocusing on a software solution.

5.5 - Privacy

At its current stage, the solution hosts 100% of users' data on their local device. Thus, there is little risk of privacy leakage due to the use of the Smart Squat solution. As the solution expands to hosting user data to further improve the accuracy of the model and user experience, the risk inherent to possessing user health data also grows rapidly. The following are a list of anticipated privacy considerations and the steps the Smart Squat solution will take to mitigate them:

- **Data leak**: The storage of user payment and health data exposes the solution to hacking attempts. As opposed to hosting the solution on company owned servers, which would require the team to maintain the solution's resiliency and security, the solution would likely be hosted in the cloud by a popular cloud service provided, such as AWS or Microsoft Azure. These cloud solutions provide many layers of security and infrastructure resiliency that a small-scale organization cannot match. Additionally, to further secure the data held, the solution can encrypt all user data as it enters the cloud infrastructure.
- **User data usage**: As the algorithm is improved to utilize more personalized user data, it is all important that every privacy and data regulation local to the area of use is followed. This solution would likely be released to the US and Canadian markets as a beachhead. Thus it would have to adhere to the following regulations [19]
    - **Canadian Market:**
        - *Personal Information Protection and Electronic Documents Act 2000*
        - *Personal Information Protection Act, SA 2003 c P-6.5* (British Columbia)
        - *Act to modernize legislative provisions as regards the protection of personal information, 2021, Chapter 25* (Quebec)
    - **American Market**:
        - The US currently does not have a single, comprehensive data privacy and protection law. But instead contains a patchwork of State regulations [20].

Nonetheless, the Smart Squat, if taken to market will abide by every local regulation to ensure availability of the service across America.

Some notable aspects of ensuring consumer confidence in their data is to give users the following rights:

- **The right to know**, what data is being collected and stored, how it is being used whether for the core service or for marketing, and what steps are being taken to protect it.
- **The right to decide**, how their data is used. To ensure this right, all functions in which data is being used will be opt-in as opposed to opt-out. This means that user data will only be collected and used if the user themselves consent.
- **The right to be forgotten,** at any time point in time. The user should be able to query the data collected from them and request it be permanently deleted.

The conferring of these rights aligns the Smart Squat solution with the EU *General Data Protection Regulation (GDPR)*, the strongest privacy and security law in the world [21]. Not only does this ensure the solution adheres to the highest standards. It also primes the solution for entering the European market and other markets with less stringent privacy laws.

6.0 – Compliance with Specifications

The team has met most of the functional and interface requirements of the system. The team opted not to use any hardware and chose to use a laptop instead of a touch screen display. The application can detect the user's movements, analyze their form in real-time, and provide feedback without disturbing the user. The performance requirements are also met as the system can analyze the user's form within the specified time range. Below is the hardware table the team used in the planning stages of the blueprint.

Table 5 - Compliance with Performance Specifications

	Specification	**Target Value**	**Tolerance**

RGB Camera	Motion capture and skeletal detection of the user, should communicate visual information directly to the embedded computer.	95% Accuracy	+-10%
Embedded Computer	Computer should be able to process data and output performance results based on model.	RAM 4G - 8G	+- 1G
Mount (Tripod)	Device should be able to reliably position the camera at the same height every time.	1 meter	+- 50 cm
Display	Device will display form score and if the squat was properly executed clearly and legibly.	480p aspect ratio	+- 200 pixels
Power Source	Battery should last for the period that the user is performing their squats without recharging. Can use a wired power supply only if necessary	Battery life: 1-2 Hours	+-1 Hour

In terms of software specifications, below is the blueprint table used to determine if the team met the goals set for the vision of the finished Smart Squat application. Detailed explanation and small changes will be explained in this section.

Table 6 - Compliance with Functional Requirements

1	**Functional requirements**	**Specifications Met? (Yes/No)**
1.1	Detect when a user is standing in front of it to record their movements. Feed that movement data to a machine learning model that can output an affirmation saying the technique is good as well as giving offer advice on how to improve form, all in real-time.	Yes, the application records the user's movements once they press record. That movement can be predicted accurately. No, the application cannot give advice yet on how to improve form but gives the user a grade based on angle calculation of their joints using the

		skeleton overlay, that can give them a idea on if they were leaning too far forward, feet were in the right position, etc.
1.2	Use a neural network for real time object detection to construct a skeletal frame of the user. Deliver the ML model's feedback in a way that does not disturb the user.	Yes, real time joint detection was achieved for the skeleton overall, and the ML model using MoViNet can give a prediction on the user's movement without disrupting the user's workout.
2	**Interface requirements**	
2.1	The display will show user performance scores for the squat exercise. A touch screen display will help the user troubleshoot the application if any problems arise while running the machine learning prediction model. The information shown on the display should be concise and ascetically pleasing.	Yes, the application gives feedback to the user in terms of quality ranges for each category and displays it on the screen. This implementation was done without the touch screen implementation a laptop can be used.
2.2	It is not required for the user to be able to install the software, it should be preloaded onto the embedded computer and should run on power up. A simple interface menu will pop up on the screen and will ask the user if they would like to start their squatting session.	Yes, without the embedded computer as we chose not to use any hardware. The application has a straightforward interface that can be activated anywhere at any time using the python user interface can start at a press of a button.
3	**Performance requirements**	
3.1	Real time analysis of user's performance of the squat within a range of 10-90 seconds.	Yes, able to analyze form based on joint detection and angle calculations within 10 -90 seconds depending how long the user records themselves for.

3.2	Setup of the hardware should be straight forward and easy to use. Turning on the embedded computer with one push of a button, and afterwards adjusting the mount to fit the frame and height of the user's body during the squat.	Yes, hardware was not used, however the laptop webcam can be adjusted to fit the user's frame by having it set up farther away and is easy to use once setup on the machine with our intuitive user interface.

Overall, the system specification goals the team set in the blueprint were met with slight adjustments to fit the lack of hardware used. Software specifications are almost all met without considering system integration of the machine learning model and skeleton overlay calculations.

7.0 – Conclusions and Recommendations

The main technical lessons learned from the project involve how to use OpenCV and 3D CNN machine learning models. This project highlighted the importance of finding a quality dataset when training machine learning models and utilizing available industry resources rather than doing everything from the ground up. This is especially true in an environment where time and money constraints are tight.

If the project were to be completed again from the start, there are key recommendations that would make it more successful. First, the development phase for all subsystems should have begun earlier. This would have allowed issues regarding the speed and accuracy of the original classifier to be identified earlier. The pivot towards the MoViNet model would have happened earlier, and there would have been far more time to integrate all 3 subsystems together. The team should have identified a more concise minimum viable product and refrained from over-engineering the prototype. For example, hosting a full stack web application on AWS using React, a database, and middleware greatly slows down development. This is because getting local environments working on different machines can be unreliable or time consuming. It also forces team members to learn skills that are not explicitly related to computer vision and convolution neural networks, the key learning outcomes of the project. If a more streamlined Python application was selected from the start, the team would have had more time to focus on UI improvements and fine tuning the feedback system.

Further research is encouraged regarding how different body mechanics can impact squat form. Lifters with longer femur lengths relative to their tibia will naturally lean forward more. Without this adaptation, they would fall backwards. Additionally, factors such as femoral neck angle and hip socket depth can influence squat mechanics [22]. This makes it such that proper form for one person may not be proper for another. Further research regarding these proportions would allow the application to be tailored to different lifters once they input their measurements. In a more ambitious approach, the program could calculate these ratios and measurements like how joint angles are currently calculated. Another medium to investigate is wearable technology. Researchers in the Harvard Bio Design Lab have developed wearable devices capable of monitoring strength training by identifying and measuring weight exercise routines. They work by incorporating artificial intelligence, that output an individualized strength score to the wearer during a wide range of strength training modalities to help manage individualized performance goals and assess progress [19]. The project began during the summer of 2022 and is listed as one of Harvard's emerging startups. Though initially cast aside by group 52 due to added weight and reduced accessibility, wearable technology could help supplement computer vision systems to increase accuracy and quality of feedback. It is unlikely that users would need a feedback system for every set of their workout. It would likely be used sporadically to check-in on their form (similar to how personal trainers are often hired once a week).

Another factor to re-evaluate is the importance of the exercise classifier entirely. Between workouts, lifters often must change their weights, stretch, or get water. This makes it such that manually selecting an exercise for feedback takes a relatively short period of time. If the classification algorithm takes too long to classify an exercise, it may make more sense for the user to manually select the exercise from a menu as they know which one will be completed. For example, if the user knows that they are performing a bench press, it is easier for them to tell the application that information than for the application to spend the first few reps making that estimate.

There are many factors to consider with respect to commercialization and manufacturing on a mass scale. The project being free of physical components helps with this. However, it is unlikely a desktop application has commercial appeal, as gym-goers do not bring their laptop. The current classification algorithm is too intensive with computer resources to be feasible. Due to these two factors, the application should be migrated to a mobile web application using an iPhone or Android camera. Hosting deep learning applications on the cloud boasts many advantages. GPUs and CPUs can be clustered which reduces training and classification speed. Workloads can be distributed and scaled at will to match peak times. AWS is very effective for hosting deep learning algorithms on the cloud, especially with computer vision [20]. To offset this cost, the application would require ads, a subscription fee, a one-time fee, or selling user data [21]. Of these options, a subscription fee is the least invasive to consumers, and ensures a consistent stream of revenue relative to the amount of total users.

Another option for commercialization is to house the application in dedicated hardware like industry competitors Tempo and Tonal [7][8]. To differentiate, the device could be mounted on existing mirrors, which would keep the cost lower and in line with the original philosophy of keeping the technology accessible. Current size for smart fitness technology is estimated to be $45 Billion USD [23]. Most of this segment consists of wearable technology. However, other hardware based fitness startups have been successful such as Mirror which sold to Lululemon for $500 Million in 2020 [26].

As mentioned earlier in the report, the application could have the benefit of making personal training more accessible to the financially disadvantaged. It is not expected to make personal trainers obsolete but could be used as a tool to supplement their teachings. To mitigate any adverse effects of the product, it will be important to review the feedback algorithm with industry professionals and test its effects on a far larger scale.

8.0 – Team Distribution of Effort

Table 7 - Team Distribution of Effort

Name	**Overall effort expended (%)**
Thierry Jones	100
Andrew Kroeger	100
Temitayo Oduyemi	100
Sean Pollen	100

References

[1] “Facts + Statistics: Sports injuries | III.” https://www.iii.org/fact-statistic/facts-statistics-sports-injuries (accessed Apr. 09, 2023).

[2] “National Electronic Injury Surveillance System (NEISS),” *U.S. Consumer Product Safety Commission*. https://www.cpsc.gov/Research--Statistics/NEISS-Injury-Data (accessed Apr. 09, 2023).

[3] U. Aasa, I. Svartholm, F. Andersson, and L. Berglund, “Injuries among weightlifters and powerlifters: a systematic review,” *Br. J. Sports Med.*, vol. 51, no. 4, pp. 211–219, Feb. 2017, doi: 10.1136/bjsports-2016-096037.

[4] D. I. Melton, J. A. Katula, and K. M. Mustian, “The Current State of Personal Training: an Industry Perspective of Personal Trainers in a Small Southeast Community,” *J. Strength Cond. Res. Natl. Strength Cond. Assoc.*, vol. 22, no. 3, pp. 883–889, May 2008, doi: 10.1519/JSC.0b013e3181660dab.

[5] C. B. C. News ·, “No regulation of personal trainers | CBC News,” *CBC*, Sep. 27, 2002. https://www.cbc.ca/news/canada/no-regulation-of-personal-trainers-1.319454 (accessed Apr. 09, 2023).

[6] M. L. Sartore and G. B. Cunningham, “Weight Discrimination, Hiring Recommendations, Person–Job Fit, and Attributions: Fitness-Industry Implications,” *J. Sport Manag.*, vol. 21, no. 2, pp. 172–193, Apr. 2007, doi: 10.1123/jsm.21.2.172.

[7] “Award-Winning AI-Powered Home Gym Membership,” *Tempo*. https://tempo.fit (accessed Apr. 09, 2023).

[8] “Tonal | The World’s Smartest Home Gym Machine For Strength & Fitness,” *Tonal*. https://www.tonal.com/ (accessed Apr. 09, 2023).

[9] D. Kondratyuk, and Y. Liangzhe, “MoViNets: Mobile Video Networks for Efficient Video Recognition,” Google, Apr. 2021. [Online]. Available: https://arxiv.org/abs/2103.11511

[10] J. Carreira, E. Noland, and A. Banki-Horvath, “A Short Note about Kinetics-600,” DeepMind, Aug. 2018. [Online]. Available: https://arxiv.org/abs/1808.01340

[11] K. Nosaka and M. Newton, “Difference in the magnitude of muscle damage between maximal and submaximal eccentric loading,” *J. Strength Cond. Res.*, vol. 16, no. 2, pp. 202–208, May 2002.

[12] “Optimizing Squat Technique : Strength & Conditioning Journal.” https://journals.lww.com/nsca-scj/Abstract/2007/12000/Optimizing_Squat_Technique.1.aspx (accessed Apr. 09, 2023).

[13] S. Lorenzetti *et al.*, “How to squat? Effects of various stance widths, foot placement angles and level of experience on knee, hip and trunk motion and loading,” *BMC Sports Sci. Med. Rehabil.*, vol. 10, no. 1, p. 14, Jul. 2018, doi: 10.1186/s13102-018-0103-7.

[14] K. Soomro, “UCF101 - Action Recognition Data Set,” University of Central Florida, Oct. 2013. [Online]. Available: https://www.crcv.ucf.edu/data/UCF101.php#Results_on_UCF101

[15] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, “OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 172–186, Jan. 2021, doi: 10.1109/TPAMI.2019.2929257.

[16] “77 Gym Membership Statistics, Facts, and Trends [2020/2021],” *Athletic shoe reviews*. https://runrepeat.com/gym-membership-statistics (accessed Apr. 10, 2023).

[17] “Fitness Trainers and Instructors : Occupational Outlook Handbook: : U.S. Bureau of Labor Statistics.” https://www.bls.gov/ooh/personal-care-and-service/fitness-trainers-and-instructors.htm#tab-6. (accessed Apr. 10, 2023).

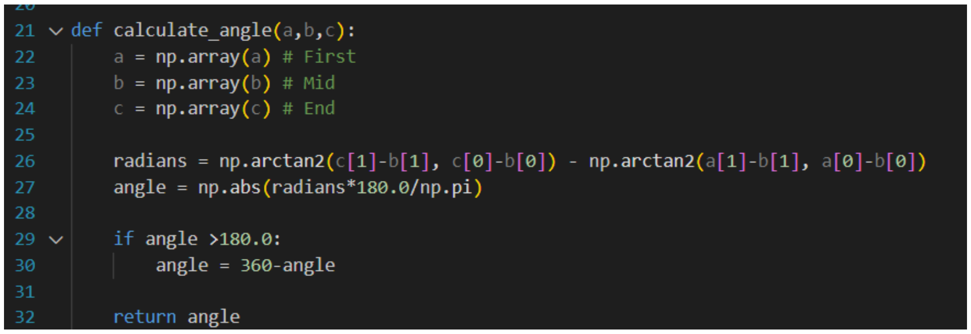
[18] “New – Customer Carbon Footprint Tool | AWS News Blog,” Mar. 01, 2022. https://aws.amazon.com/blogs/aws/new-customer-carbon-footprint-tool/ (accessed Apr. 10, 2023).

[19] “Canada - Data Protection Overview,” *DataGuidance*, Feb. 01, 2022. https://www.dataguidance.com/notes/canada-data-protection-overview (accessed Apr. 10, 2023).

[20] G. A. Canada, “United States – Comprehensive state privacy laws,” *GAC*, Jun. 20, 2022. https://www.tradecommissioner.gc.ca/guides/state_privacy_laws_lois_confidentialite.aspx?lang=eng (accessed Apr. 10, 2023).
[21] “The general data protection regulation,” Sep. 01, 2022. https://www.consilium.europa.eu/en/policies/data-protection/data-protection-regulation/ (accessed Apr. 10, 2023).
[22] “How Your Anatomy Influences Your Squat Mechanics,” *JC Fitness*. https://jcfitness.co.uk/blog/how-anatomy-influences-squat-mechanics/ (accessed Apr. 10, 2023).
[23] “Wearables to Monitor Strength Training,” *Harvard Office of Technology Development*. https://otd.harvard.edu/explore-innovation/technologies/wearables-to-monitor-strength-training/ (accessed Apr. 10, 2023).
[24] “Deep Learning on AWS,” *Amazon Web Services, Inc.* https://aws.amazon.com/deep-learning/ (accessed Apr. 10, 2023).
[25] G. Appel, B. Libai, E. Muller, and R. Shachar, “On the monetization of mobile apps,” *Int. J. Res. Mark.*, vol. 37, no. 1, pp. 93–107, Mar. 2020, doi: 10.1016/j.ijresmar.2019.07.007.
[26] S. Maheshwari, “Lululemon to Buy Mirror, a Fitness Start-Up, for $500 Million,” *The New York Times*, Jun. 29, 2020. Accessed: Apr. 10, 2023. [Online]. Available: https://www.nytimes.com/2020/06/29/business/lululemon-buys-mirror.html

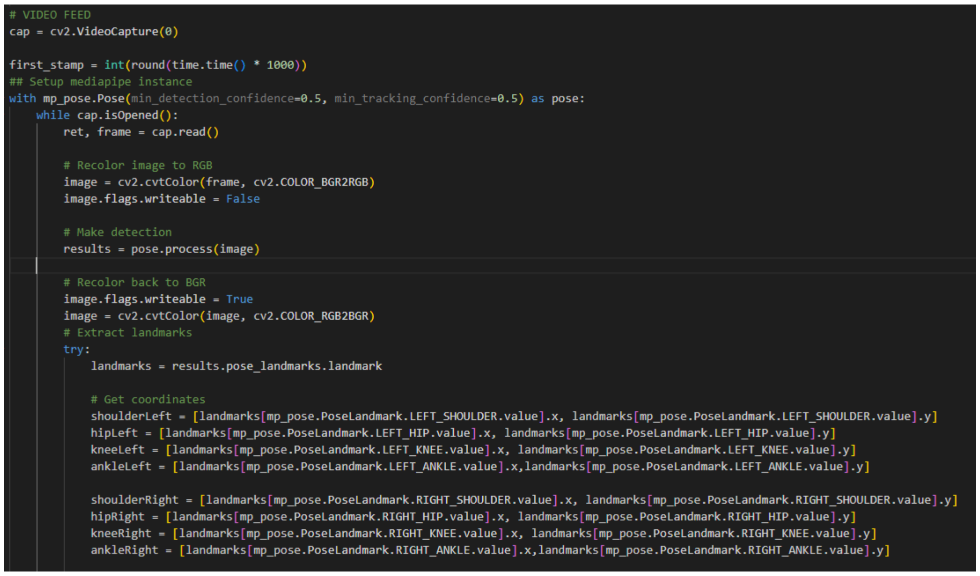
Appendix

Skeleton Overlay Code Snippets



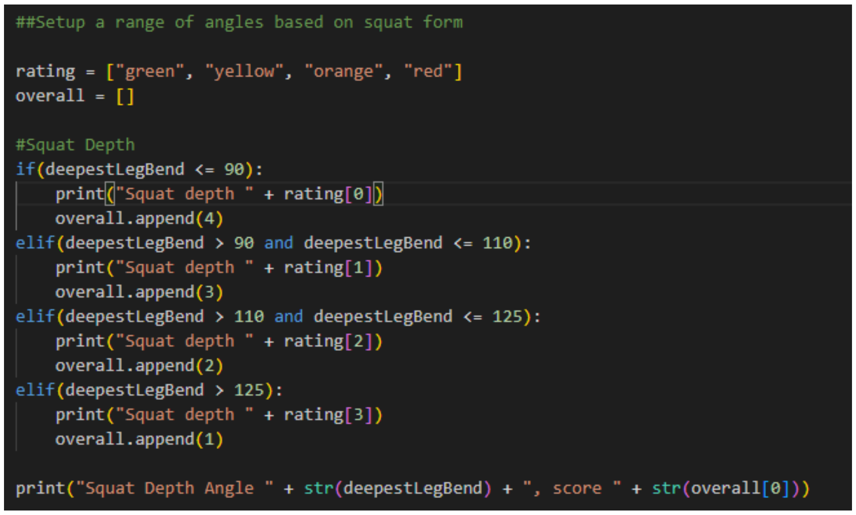
```python
def calculate_angle(a,b,c):
    a = np.array(a) # First
    b = np.array(b) # Mid
    c = np.array(c) # End

    radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[0]-b[0])
    angle = np.abs(radians*180.0/np.pi)

    if angle >180.0:
        angle = 360-angle

    return angle
```

Figure 11- Calculate Angle Function



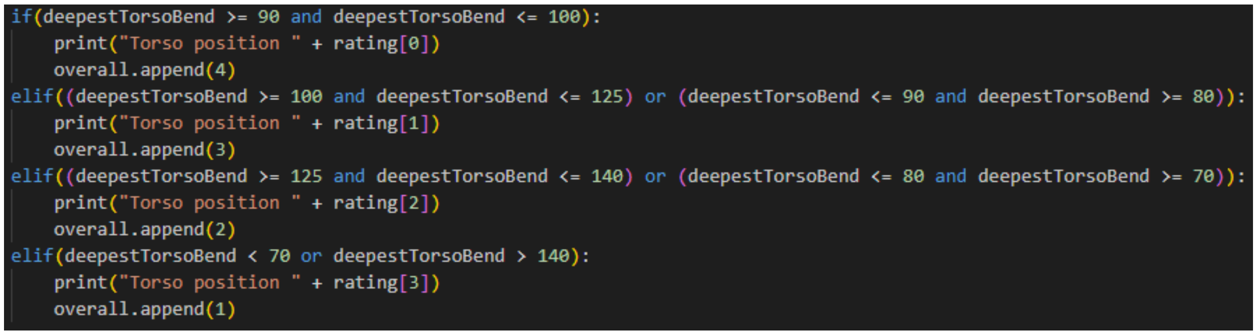
```python
# VIDEO FEED
cap = cv2.VideoCapture(0)

first_stamp = int(round(time.time() * 1000))
## Setup mediapipe instance
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
    while cap.isOpened():
        ret, frame = cap.read()

        # Recolor image to RGB
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Make detection
        results = pose.process(image)

        # Recolor back to BGR
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        # Extract landmarks
        try:
            landmarks = results.pose_landmarks.landmark

            # Get coordinates
            shoulderLeft = [landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x, landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
            hipLeft = [landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x, landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].y]
            kneeLeft = [landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].x, landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].y]
            ankleLeft = [landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value].x,landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value].y]

            shoulderRight = [landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].x, landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].y]
            hipRight = [landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].x, landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].y]
            kneeRight = [landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].x, landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].y]
            ankleRight = [landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value].x,landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value].y]
```

Figure 12- Video Capture and Joint Coordinates



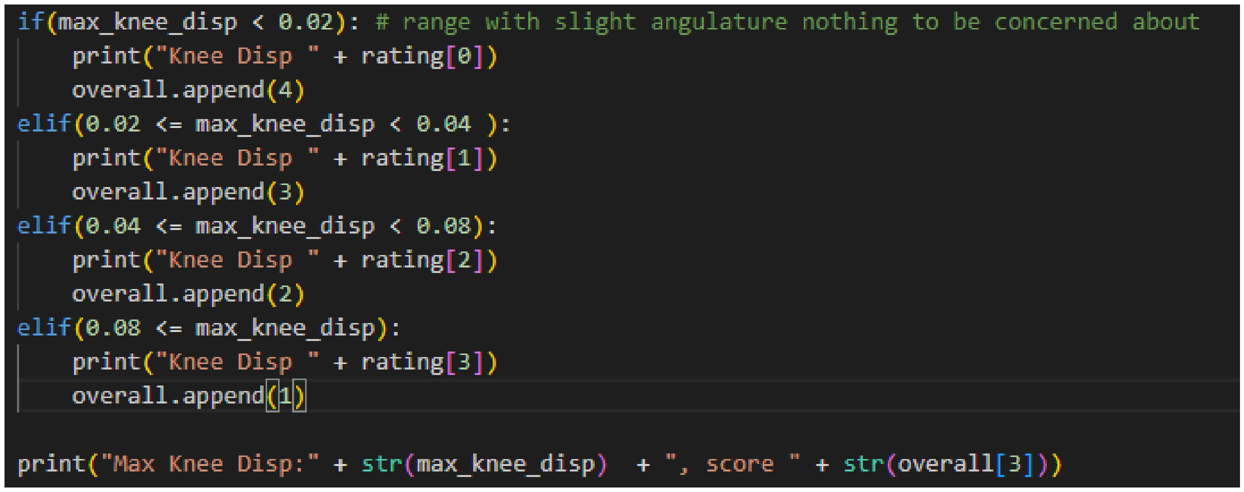
```
##Setup a range of angles based on squat form

rating = ["green", "yellow", "orange", "red"]
overall = []

#Squat Depth
if(deepestLegBend <= 90):
    print("Squat depth " + rating[0])
    overall.append(4)
elif(deepestLegBend > 90 and deepestLegBend <= 110):
    print("Squat depth " + rating[1])
    overall.append(3)
elif(deepestLegBend > 110 and deepestLegBend <= 125):
    print("Squat depth " + rating[2])
    overall.append(2)
elif(deepestLegBend > 125):
    print("Squat depth " + rating[3])
    overall.append(1)

print("Squat Depth Angle " + str(deepestLegBend) + ", score " + str(overall[0]))
```

Figure 13 - Squat Depth Ranges/Ratings



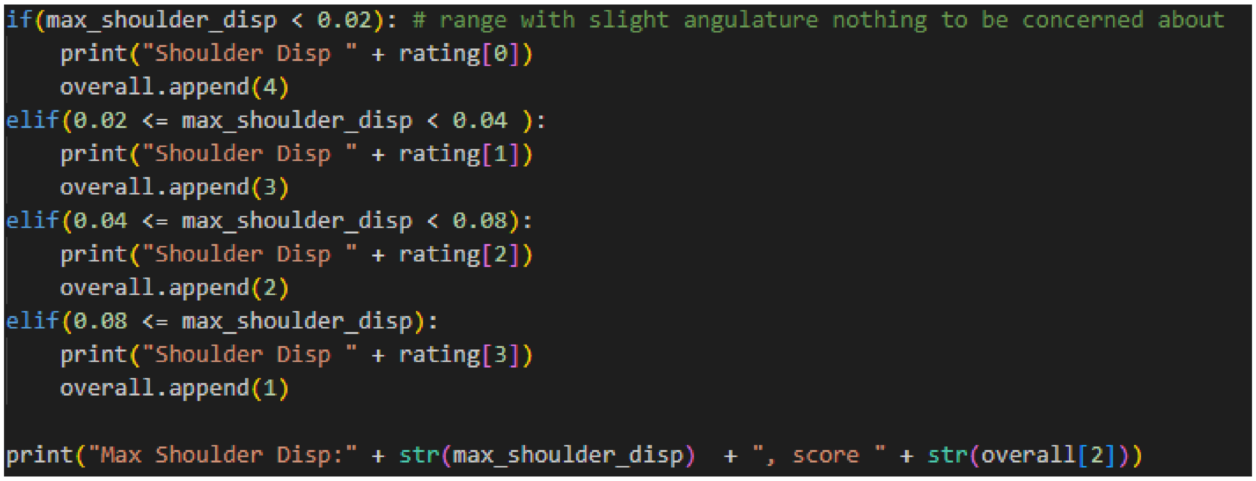
```
if(deepestTorsoBend >= 90 and deepestTorsoBend <= 100):
    print("Torso position " + rating[0])
    overall.append(4)
elif((deepestTorsoBend >= 100 and deepestTorsoBend <= 125) or (deepestTorsoBend <= 90 and deepestTorsoBend >= 80)):
    print("Torso position " + rating[1])
    overall.append(3)
elif((deepestTorsoBend >= 125 and deepestTorsoBend <= 140) or (deepestTorsoBend <= 80 and deepestTorsoBend >= 70)):
    print("Torso position " + rating[2])
    overall.append(2)
elif(deepestTorsoBend < 70 or deepestTorsoBend > 140):
    print("Torso position " + rating[3])
    overall.append(1)
```

Figure 14 - Torso Angle Ranges/Ratings



```
if(max_knee_disp < 0.02): # range with slight angulature nothing to be concerned about
    print("Knee Disp " + rating[0])
    overall.append(4)
elif(0.02 <= max_knee_disp < 0.04 ):
    print("Knee Disp " + rating[1])
    overall.append(3)
elif(0.04 <= max_knee_disp < 0.08):
    print("Knee Disp " + rating[2])
    overall.append(2)
elif(0.08 <= max_knee_disp):
    print("Knee Disp " + rating[3])
    overall.append(1)

print("Max Knee Disp:" + str(max_knee_disp)  + ", score " + str(overall[3]))
```

Figure 15 - Knee Displacement Ranges/Ratings



```
if(max_shoulder_disp < 0.02): # range with slight angulature nothing to be concerned about
    print("Shoulder Disp " + rating[0])
    overall.append(4)
elif(0.02 <= max_shoulder_disp < 0.04 ):
    print("Shoulder Disp " + rating[1])
    overall.append(3)
elif(0.04 <= max_shoulder_disp < 0.08):
    print("Shoulder Disp " + rating[2])
    overall.append(2)
elif(0.08 <= max_shoulder_disp):
    print("Shoulder Disp " + rating[3])
    overall.append(1)

print("Max Shoulder Disp:" + str(max_shoulder_disp)  + ", score " + str(overall[2]))
```

Figure 16 - Shoulder Displacement Ranges/Ratings

Machine Learning Code Snippets

```
model = load_movinet_from_hub('a2', 'base', hub_version=3)
```

```
for item in os.listdir('vids/'):
  mp4 = 'vids/' + item
  gif = 'vidsGif/' + item.replace('mp4', 'gif')
  clip = VideoFileClip(mp4)
  clip = clip.set_fps(5)
  clip.write_gif(gif)
```

```
for item in sorted(os.listdir('vidsGif/')):
  gif = 'vidsGif/' + item
  video = load_gif(gif, image_size=(224, 224))
  media.show_video(video.numpy(), fps=30)
  outputs = predict_top_k(model, video)

  for label, prob in outputs:
    print(label, prob)
```

Budget (From Blueprint)

Table 8 - Original Blueprint Bill of Materials

Component	Description	Supplier	Units	Estimated Cost
OAK-D lite RGB Camera [5]	13MP primary camera, 480p with depth, 120FPS, 0.6-4.5W power consumption.	Mouser Electronics Canada [6]	1	*$205.62*
Raspberry Pi 4 [7]	Quad Core 64 Bit, with WiFi and Bluetooth capability. 4GB of RAM.	Amazon.ca	1	$239.99 product + $31.19 taxes *$271.19*
Power Supply [8]	USB-C 5.1V 3A. Designed specifically for Raspberry Pi 4.	Amazon.ca	1	$19.95 product + $2.59 taxes *$22.54*
Touch Screen [9]	3.5-inch display, 320x480 resolution, LCD touch screen, includes fan. Designed specifically for Raspberry Pi 4.	Amazon.ca	1	$35.98 product + $4.68 taxes *$40.66*
				Total: *$540.01*

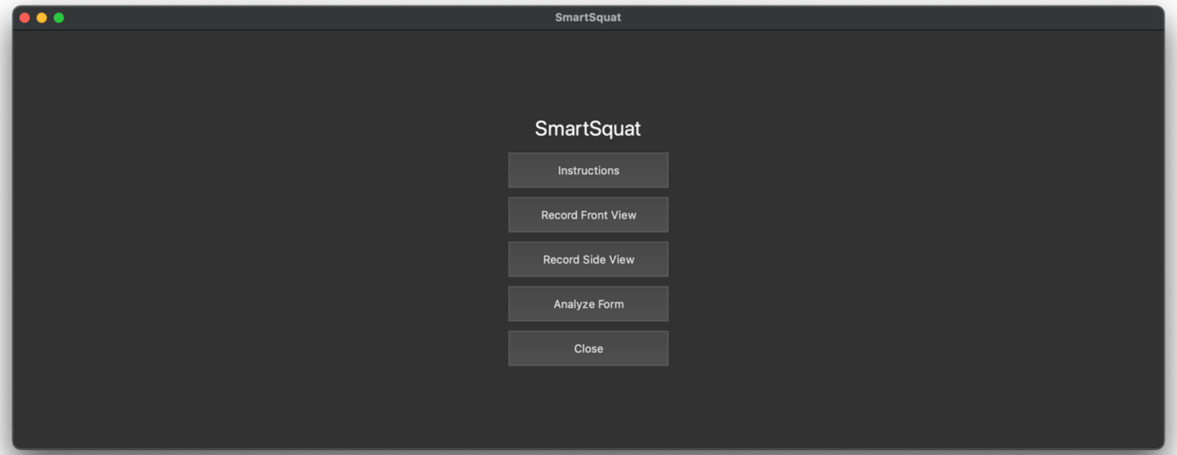


Figure 17 - Original UI Colour Scheme