# From Flab to Ab: A Smart Home Gym

Jake Betenson, Matthew Cranford, Cole Jacobs, Tyler Linquata, and Ryan Lukas

*Abstract*— **Physical exercise is a critical component of any healthy lifestyle. For some people, formal exercise – especially when it includes weight training – can be intimidating. For this project, we built a virtual fitness coach that provides interactive feedback on the user's movements and form. This tool is a "smart mirror" – a two-way mirror with an electronic display behind the glass. Other features include automatic workout logging to personalize the users experience. The information gathered and generated for the user will also be made accessible through a mobile application for added convenience and availability.**

## I. INTRODUCTION

INDIVIDUALS trying to improve their health often perceive exercise and fitness to be a daunting task. The anticipation and intimidation they feel may lead further to frustration or feeling discouraged about their fitness goals. Many of these concerns are legitimate – if one does not know and use proper form, the risk of exercise-related injury increases. To avoid this, many seek help from a personal trainer. With professional instruction, individuals learn to use correct form and how to structure an effective exercise schedule. Not only is the risk of injury reduced, but also the results of the workout become more pronounced.

Though there are many benefits associated with receiving personal training, there are also drawbacks. Some individuals may find the high, recurring price to be an obstacle to their fitness goals. Others may find the advanced planning and rigid schedule to be challenging. Our solution to these problems was to create a personal trainer without the person.

Using modern technology, we built a *virtual* trainer that can replace the need for a personal trainer and provide benefits to both beginners and veterans of fitness. This trainer takes the form of a "smart mirror" – a two-way mirror with an electronic display behind the glass. The finished product is a tool that removes much of the pain of working out.

### A. Project Synopsis

The core of this project is the smart mirror. Mirrors are commonly used in gyms as a way to provide positive reinforcement and motivation to exercisers. They also allow one to see exactly what the body is doing, which is helpful when practicing new movements. A smart mirror retains these functions, but it also provides the opportunity to go beyond a simple reflection.

[1]The authors are with the Electrical and Computer Engineering Dept., University of Utah, Salt Lake City, UT 84112 USA (email: j.betenson@utah.edu, u1147580@utah.edu, cole.jacobs@utah.edu, tyler.linquata@utah.edu, u1063988@utah.edu)

A smart mirror is typically composed of three components: two-way glass, an electronic display, and a computing device. Two-way glass is designed to reflect light from one direction while allowing light to escape from the other. Placing the electronic display behind the glass enables the reflection typically seen in a mirror to be joined by images projecting from the screen. By connecting the electronic display to the computing device, one can customize the display with graphical widgets, providing an interactive experience.

This project's smart mirror is comprised of plexiglass sheet with a mirror coating that is encased in an attractive frame, which also houses an Nvidia Jetson and an LCD television. A camera is mounted to the frame to allow video capture of the space immediately in front of the mirror. These components, running appropriate software, constitute the virtual trainer. The trainer provides feedback on the user's form with the use of a overlay on the user's reflection with lines indicating current position.

### B. Demonstration

The demonstration showcases the core features of the project: the smart mirror, the mobile application, and the smart trainer. This will require sufficient space to fit the mirror, exercise equipment, and room to perform the exercises. The demonstration requires 20 square feet, in a relatively simple environment.

Over the course of the demonstration, attendees can perform a supported exercise, such as a squat, in front of the mirror in order to get feedback regarding their exercise form and their number of exercise repetitions. This information is gathered by camera and processed on the mirror's attached Nvidia Jetson Nano.

Most saliently, the display will include a skeletal tracker to assist with exercise form. During the exercise, a video feed is displayed on the mirror with the skeletal tracker overlaid. This video feed also includes graphics to help the user understand how the exercise should be performed.

## II. BACKGROUND

### A. Human Pose Estimation

This project is strongly influenced by ongoing research related to body pose estimation. Body pose estimation refers to the process of estimating a person's pose by inferring where body joints are. Its possible applications are many, including being used to monitor and analyze exercise form.

Beginning with "DeepPose" by Toshev et al. [1], efforts in human pose estimation began utilizing Deep Neural Networks (DNNs) and other machine learning techniques to better approximate joint locations. The speed and accuracy of these systems continue to improve with advancements in neural network configurations. One particularly useful resource for our project will be OpenPose [2]. This is a pre-trained real-time system available as an open source library for research purposes. This library utilizes OpenCV [3] to associate body parts with provided images and video. A similar library, TRTPose, can be run on the Nvidia Jetson to analyze posture and movement during exercise.

To guide our definition of proper form, we will primarily refer to academic research conducted on sports biomechanics and strength and conditioning [4], [5]. These sources provide mathematical descriptions of lifting technique that will be critical for building our model of proper form.

### B. Related Work

This project belongs to a growing trend of fitness-related startups and smart gym equipment. In 2019, venture capitalists spent hundreds of millions of dollars on fitness startups in the United States [6]. We are clearly not alone in our excitement for technology in this market.

Though not yet prevalent for the average consumer, there are several products which offer features that overlap with our proposed project. The two products most similar to our own are Tempo [8] and MIRROR [7]. The latter of these products can be seen in Fig. 1. Both of these products use screens which are designed to be looked at during exercise and use sensors to count reps and analyze form – features we would like to emulate. However, these products differ from the proposed project in their focus on live, subscription-based training classes. No such content will be provided in this project.

We are separated in other ways from these products including in the amount of money available to us and in our lack of privacy concerns. Since this is an academic pursuit, our budget is limited. We will also be making use of open source technologies that are available to us under the condition that we do not profit on our project.

Fig. 1. Example of Mirror Co. product [7]

## III. Design Overview

### A. Frontend

The frontend consists of GUI written in python that displays the informed image, current exercise, the current rep count, and workout history to the user. The image has an overlay that helps the user understand how to perform the exercise. The overlay will change color to indicate 'how close' the user is to finishing one repetition as well. This direct feedback gives the user an intuitive feel of the workout and whether they are performing it both correctly and safely.

Switching to a GUI based option was not our first choice. Originally we had plan to use Node.js and have a web server display the image. This way we could manipulate the screen and make the mirror look and feel like a mirror. Unfortunately, pose-estimation on video streams in real time is resource intensive and the through-put of images to the web server was too slow to produce any usable feedback. Our frames-per-second (FPS) went from approximately 13fps to less than 2fps, making the web server idea a dead end. That is not to say that the idea is not possible but with our limited resources, pivoting to a different framework was more suitable.

After choosing to build our GUI with python, we chose the library PySimpleGui after iterating through different python GUI libraries. An issue we encountered while developing the frontend is the lack of documentation of the GUI library used. The lack of documentation made it difficult to know exactly how to manipulate the provided code to produce exactly what was designed. For example, a certain function call was necessary for the GUI to update its state but the call was deceptively unclear that it was needed (it was called window.read() ), but nothing would function correctly unless this line of code was kept.

Lastly, our final challenge was resizing the GUI to make sure everything looked adequate. We tried expanding the window by setting the size to fit the monitor (720p) but only filled up about one half of the screen. After playing around with the setting, we made sure to set the output display of the jetson to 720p. We also realized that the drawn image coming from the Jetson can not be resized because we would drastically loose FPS. Therefore, we made the drawn image that's being pipe-lined in one half of what the final GUI size would be. Pipe-lining the image at the specific size solved the problem of having to resize the image once it gets to GUI which helped retain our current FPS.

### B. Backend

The first essential step in developing the backend is configuring the development environment correctly and in an orderly manner. This proved to be quite a challenge with the Jetson Nano for several reasons namely, there are many pose-estimation frameworks standards that rely on various different versions of python and its associated libraries. This meant that when attempting to test out a specific framework, it would interfere with other installations. Another issue is that most of the current pose-estimations frameworks are

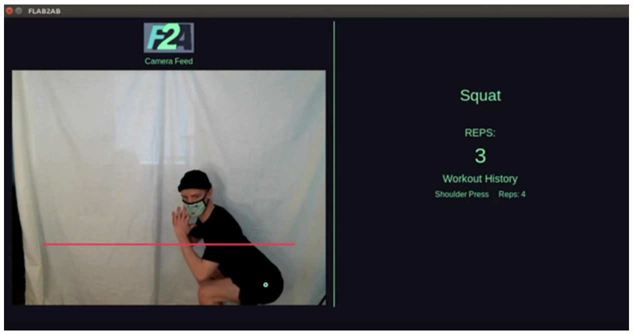


Fig. 2. Final GUI Application

not optimized for the Jetson Nano and as such performed poorly when tested, providing an undesirable frame rate of 2-3 fps. This issue was solved when we began using virtual environments to keep installations isolated from each other and discovered a framework that had been optimized exactly for the Jetson Nano.

With the correct libraries and frameworks in place, a true Integrated-Development-Environment (IDE) was lacking. The initial hope was that the native Ubuntu operating system on the Jetson would allow us to have some nice tools for remote development. This hope was dashed when it was discovered how resource intensive the IDE was, making it impossible to simultaneously develop and test. This roadblock was overcome by utilizing some clever programming tools such as tmux. This tool allows users to share the same shell terminal making it easier to pair program in a limited environment.

In order to provide exercise feedback, the backend utilizes NVidia's TRTPose. TRTPose has several pretrained neural networks optimized for NVidia's family of Jetson products capable of human pose estimation. These networks take in a resized image from the attached webcam and parse key joints on the humans detected in the image. This implementation uses ResNet-18, an eighteen layer convolutional neural network that utilizes skip connections for faster inference and training. Compared to DenseNet-18, which was only capable of doing inference on 2-3 frames per second, ResNet-18 provides 13-15 frames per second. This level of performance is enough to provide a smooth user experience.

Using the framework described, we successfully implemented features which provide visual feedback to the user and enable auto-logging of exercises. These features are currently supported for three exercises: bicep curl, shoulder press, and squat. In this case, visual feedback refers to highlighted planes or arcs, applied in a post-processing phase, which guide the user's movement in both direction and distance. Auto-logging of exercises refers to the applications's ability to automatically detect and record rep counts, which are paired with the user's selected exercise and exported.

Visual feedback and auto-logging are related in one important way: they depend on categorizing the user's pose into one of two states. These states are designated as *up* and *down* states, and they refer to whether the user is in the high or low portion of his/her range of motion for the exercise. Categorizing the user's state depends on which exercise they are performing, but the general principles are the same.

The first important step in categorizing a user's state is identifying which joints are relevant to that exercise. In reality, many exercises (compound lifts in particular) involve multiple muscle groups and therefore multiple joint groups. However, this application takes a less rigorous approach to form checking because the alternative would entail several cameras capturing the user's movement from different angles. As a result, only joints that are immediately relevant in our isolated view of the exercise are selected. For bicep curl, this corresponds to the shoulder, elbow, and wrist joints. For squat, the knee and hip joints are important, and for shoulder press, we're interested in the user's elbow, neck, and shoulder.

After relevant joints have been identified, the next step is to define the thresholds which separate the *up* and *down* states. These thresholds are predicated on good exercise technique. For example, completing a proper curl requires the weight to start in a resting, extended position and then be brought up until the bicep is fully contracted. This movement can be quantified by calculating the angle formed between the upper arm and the lower arm with the elbow as vertex. After accounting for body mass, which restricts the possible values of these angles, the *down* state of bicep curl was defined as anything more relaxed than 120°. The *up* state was defined as anything more contracted than 50°.

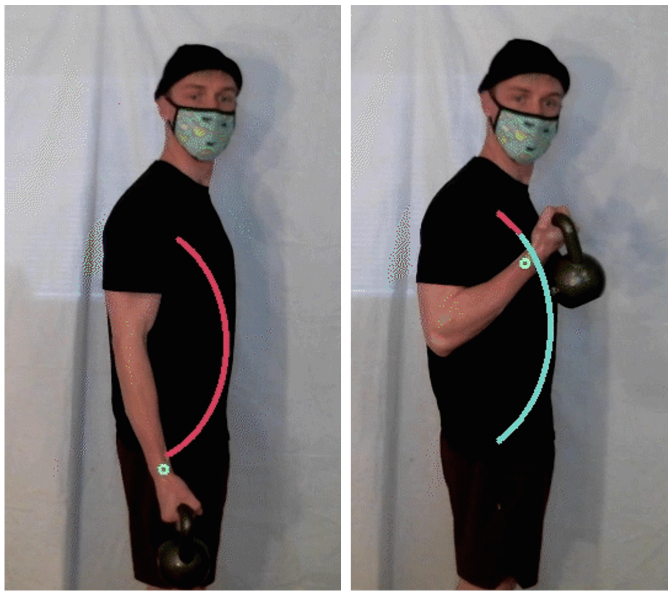


Fig. 3. Visual effects for a user performing a bicep curl

The same process can be repeated for shoulder press and squat, albeit with slightly different details. Where angles were previously used, thresholds for these exercises were more accurately defined as vertical positions relative to some body part. For squat, *up* was defined to mean that the user's hip was at least $\frac{3}{4}$ length of the femur above the knee and *down* to mean that the user's hip was no more than $\frac{1}{4}$ length of the femur above the knee. For shoulder press, *up* was defined as at least $\frac{1}{2}$ length of the forearm above the neck and *down* to mean lower than the neck. The visual feedback produced as a result represents planes which the user must cross.

With thresholds defined and the user's state properly categorized, it becomes possible to add post-processing effects to the image to communicate to the user which way he/she needs to move. These effects were applied using OpenCV [3], which exposes functions to draw lines, circles, and ellipses on a given image. Examples of visual effects for bicep curl, shoulder press, and squat are shown in Fig. 3, 4, and 5 respectively.

To implement auto-logging, the application need only track the recent history of the user's categorized states. A completed rep for any of these exercises is counted when the user successfully moves from the starting state of the exercise to the complementary state and then back to the starting state. For bicep curl and shoulder, this translates into a movement from *down* to *up* and back to *down*, whereas squat begins in the *up* state and is therefore the sequence *up* to *down* and back to *up*. This count is communicated to the phone, the frontend application, and a connected database via polling for future review.

## C. Networking

The project uses Web API calls for communication between the Jetson's backend and the Mobile Application. The Jetson's Python backend builds a dedicated HTTP server written using the Flask library. Flask is a Python-compatible micro web framework that facilitates simple web servers with custom API endpoint calls. The mobile application accesses these endpoints to command the Jetson to begin or end an exercise, get detailed knowledge about the workout, toggle debug modes, etc. The API calls include one for each exercise and additional debugging queries.

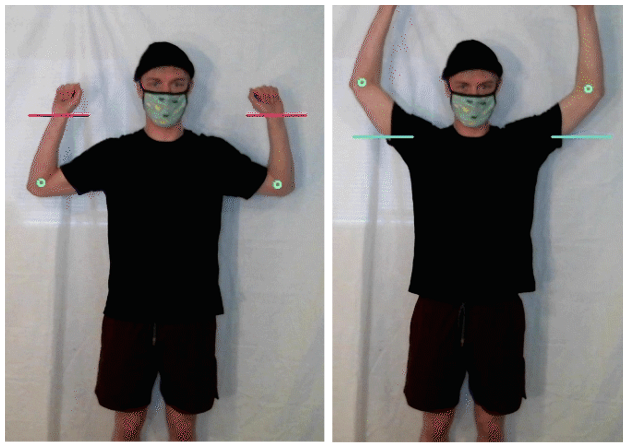


Fig. 4. Visual effects for a user performing a shoulder press

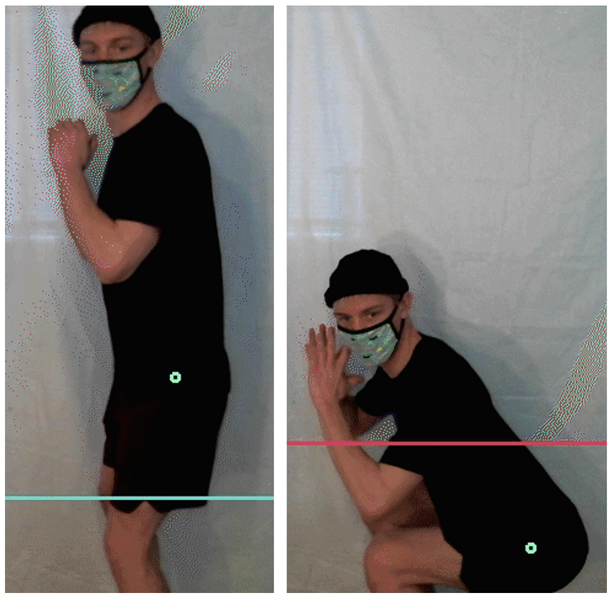
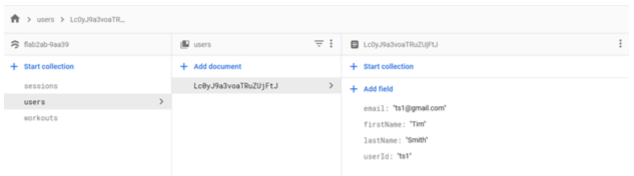


Fig. 5. Visual effects for a user performing a squat

## D. Database

Firebase is a NoSQL database that allowed us to push JSON formatted files to save and be able to retrieve information for future use. To be able to sync up with Firebase, we built functions to allow three different collections to be built: User, Session, and Workout. Each user's information would be saved such as their email, name, and a unique user identification. This unique user identification was then used as a field for each saved JSON file to be able to retrieve each specific session and workout that apply to each user.

With a NoSQL database, it is hard to retrieve the unique ID until we found out that you can pass this unique ID along with a 201 OK status to say that the individuals user profile was built. Then within the mobile application, we would hold a dictionary of keys that linked with the Firebase authentication to know which user is using the application. After the User is built within the database and the unique ID has been created, we can now add each workout session the individual takes a part of. This Session holds specifically the user identification and a unique session ID to link each workout with the session.



sessions
users
workouts

Lc0yJ9a3voaTRuZUjFtJ

email: "ts1@gmail.com"
firstName: "Tim"
lastName: "Smith"
userId: "ts1"

Fig. 6. NoSQL Firebase Database

Having a field that holds the specific session ID will allow the mobile application to hold another dictionary of each session the individual participates in. To find each workout, you would search on the session ID field to find the specific workouts they performed in the Workout collection.

This would pass back every specific workout they did and how many repetitions were performed. With three different collections, it helps keep all data separate and organized so we wouldn't have any overlapping data within a NoSQL database. This ended being a proof of concept showing that this application can hold more than one user and store multiple workout sessions for the user to go back and examine their previous workouts.

## E. Mobile Application

The mobile application is the primary method of user interaction for this device. Developed using Swift 5.0 and UIKit, the application is compatible with any iOS device, and will be compatible with upcoming MacBooks, allowing the user to control the mirror from nearly any Apple computing device.

This application was developed following the Model View Controller (MVC) design pattern. The MVC pattern was used because it is generally simple to implement and is a robust, widely used pattern in mobile app development. In the MVC pattern, code is split into the three main sections. The model represents the state of the application. It consists of any data being held by the application, as well as methods that manipulate the data. In this pattern, only the model should modify application state. The view is a group of classes that represents what the user is seeing and interacting with. Often, the view is connected to a controller, which is responsible for retrieving data for the view from the model, and for telling the model when and how it should manipulate data.

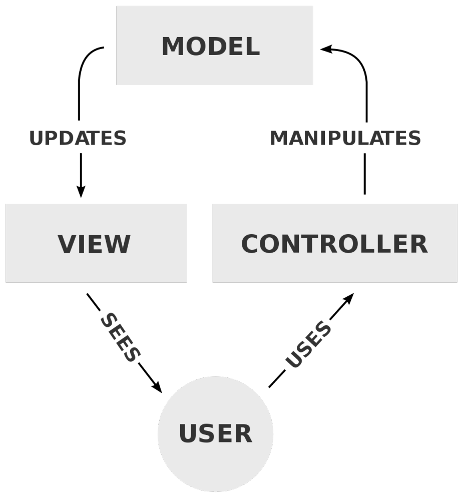


Fig. 7. Model View Controller diagram [9]

The application is capable of connecting wirelessly to the mirror and a Google Firebase instance. The API calls are handled with the Swift URLRequest and URLSession libraries. All connections to Firebase, FireAuth, and FireStore are made using the official Firebase library for Swift. All user information and authentication is securely stored on Firebase to mitigate security risks and anonymize data.

## F. Physical Construction

The mirror consists of a wooden frame that encloses a television. The frame also holds a reflective surface in place over the display. The Jetson Nano sits behind the display while a video camera is mounted on top to capture input data.

## IV. Conclusion

The current implementation allows for a video feed at 13-15 frames per second at 640x480 which is a smooth enough user experience. More powerful Jetson boards such as the TX1 or TX2 can be used to immediately improve the video feed's frames per second or resolution, but also increasing the cost. The supported exercises give basic visual feedback showing how the exercise should be performed. While the iOS application is user friendly and functions as a controller for the mirror. This is an effective baseline for future work. If development continues biometric data can be gathered from smart peripherals such as the Apple Watch or Garmin Forerunner and displayed on the mirror in real time. Microsoft's Kinect camera could be employed to get depth data from the secondary infrared camera which can be used for three dimensional exercise logic such as misaligned limbs. Lastly, we could enhance the feature suite on the mobile application with elements such as a workout history or nutrition management.

## References

[1] A. Toshev and C. Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1–9, Dec 2013.
[2] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7291–7299.
[3] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
[4] S. Lorenzetti, M. Ostermann, F. Zeidler, P. Zimmer, L. Jentsch, R. List, W. R. Taylor, and F. Schellenberg, "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 Science, Medicine and Rehabilitation*, vol. 10, no. 14, pp. 1–11, Jul 2018.
[5] P. Comfort and P. Kasim, "Optimizing Squat Technique," *Strength and Conditioning Journal*, vol. 29, pp. 10–13, Dec 2007.
[6] P. Mathur. (2019) Fitness-hardware startups follow Peloton's lead – up to a point. [Online]. Available: https://pitchbook.com/news/articles/peloton-breaks-ground-for-fitness-hardware-startups
[7] MIRROR. (2020) Mirror. [Online]. Available: https://www.mirror.co
[8] CoreTech Fitness Co. (2020) Tempo. [Online]. Available: https://tempo.fit
[9] Wikipedia. (2020) Mvc. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/a/a0/MVC-Process.svg/1200px-MVC-Process.svg.png