University of Waterloo

Faculty of Engineering

Department of Electrical and Computer Engineering

Detailed Design and Project Timeline

Smart Workout Trainer

Rupakeerthana Vemulapalli

Rajan Sood

Wen Hao Li

Lucy Han

Gerry Zhang

Group 2023.64

Consultant: Prof. Rodolfo Pellizzoni

June 22, 2022

Table of Contents

1. High-Level Description of Project

1.1 Motivation

Over the course of the COVID-19 pandemic, gym equipment purchases increased by 170%. However, with that saw a 48% rise in at-home related workout injuries [1]. Studies show over 1 in 3 injuries at fitness facilities are caused by overexertion, with improper workout posture being the leading cause. Improper form causes tension around non-targeted muscle groups, leading to decreased performance and higher risk of injury [2]. While exercising at home, the average person is not equipped to focus on their workout using proper technique. Alternatives such as hiring a personal trainer are costly, and online training resources cannot provide real-time feedback on form.

A full body length mirror is a cost-effective solution that can drastically aid in reducing risk of injury and increase safety during workouts at home by helping users observe their form [1]. Yet there lacks a safe and easy way to understand and execute upon known proper form without professional training. A smart-mirror AI trainer can be used to bridge this gap to help prevent potential injury, maximize exercise efficiency, and increase performance.

1.2 Project Objective

The objective of the Smart Workout Trainer is to design a product that monitors and corrects a user's workout posture during targeted exercises, such as weightlifting, in real-time. It uses a Kinect sensor to identify skeletal movements of the user and passes the input to an AI-model that computes and projects the proper posture onto a body-length smart mirror. The user can then use the correct skeletal projection to guide their posture in real-time.

The main advantage of this design over current alternatives is its ease of use and ability to non-intrusively recognize potential for injury in the user's body movements. Furthermore, our technology is designed to accommodate diverse body types and users, making it a more accessible, cost-effective and scalable alternative compared to hiring professional trainers or other online resources. It can also enable users to measure and track key metrics overtime.

1.3 Block Diagram

1.3.1 Hardware Subsystem

The hardware subsystem is composed of a sensor and components that make up the physical aspect of the Smart Workout Trainer. The first component is the monitor, where the user can view video playback of their critiqued workout form as well as example exercises. The next component is an IR touch frame, which enables the monitor to receive touch-screen inputs, enabling the user to interact with the program's interface to access different workout modes and settings. The last physical component of the hardware subsystem is a Kinect camera. The Kinect is the device that allows our application to capture video input from a user's movements to map out their skeletal form by sensing a user's position in x, y, z space.

1.3.2 Program Application Subsystem

The program application subsystem is the application that the user interacts with. It is the user interface that is projected onto the hardware system, allowing the user to process data and provide inputs. The program application consists of two subsystems that provide the user with data and interactive options. The first subsystem is the front-end user interface, which consists of the UI with buttons that can be selected using touch screen inputs. The second subsystem is the video processing component, which displays a video with corrections to a user's workout form. The backend service is responsible for classifying improper movements, the data is then encoded and sent to the video processing subsystem who's responsible for displaying the critiqued video.

1.3.3 Backend Service Subsystem

The backend service subsystem consists of four subsystems in total and are split into two different functionalities. The backend service aims to accomplish two tasks with these functionalities: the first being to examine the movements of a user's workout to indicate if any mistakes or improvements can be made; the second functionality is to provide a facial recognition login service using the input received from the Kinect camera.

The first subsystem is the classification algorithm that determines improper form and highlights areas of improvement for the user. The results of this subsystem are then sent to the front-end's video processing component, which will display the algorithm's feedback to the end user. The second subsystem is an AI program that computes the comparison between the user's movements and an example of a proper workout form from a database of known, proper exercises. The result of this system is analyzed using the classification algorithm mentioned previously. The third subsystem is the facial recognition login component, which uses machine learning to analyze the user's facial structure and communicates with the database to find a match and subsequently retrieve the user credentials. The last subsystem is the Kinect SDK which is the software package we leverage using the Kinect camera, this allows us to support the usage of the Kinect.

1.3.4 Database Subsystem

The database subsystem contains the necessary data and features which are fed to the backend service for processing. The database contains a table of the user's credentials for use of the facial recognition feature. The second table consists of the supported exercises as well as the decoded information that depicts the movements of the corresponding exercise.

1.3.5 User Registration Subsystem

To ensure a secure authentication system for users, a user registration subsystem is necessary to manage user credentials. There are two potential paths for creating a user registration system. The first being authentication through facial recognition. This is viable as the Kinect camera has facial recognition capabilities, which we can use to build out this system. The second path would be a temporary user session in which a user can scan a QR code to activate a user session. The first option would be persistent while the second would be temporary. The temporary solution would provide ease of use and a lighter engineering lift. However the permanent path would be more scalable and a better user experience. For our purposes we will be weighing out these two options for the scope of our project.

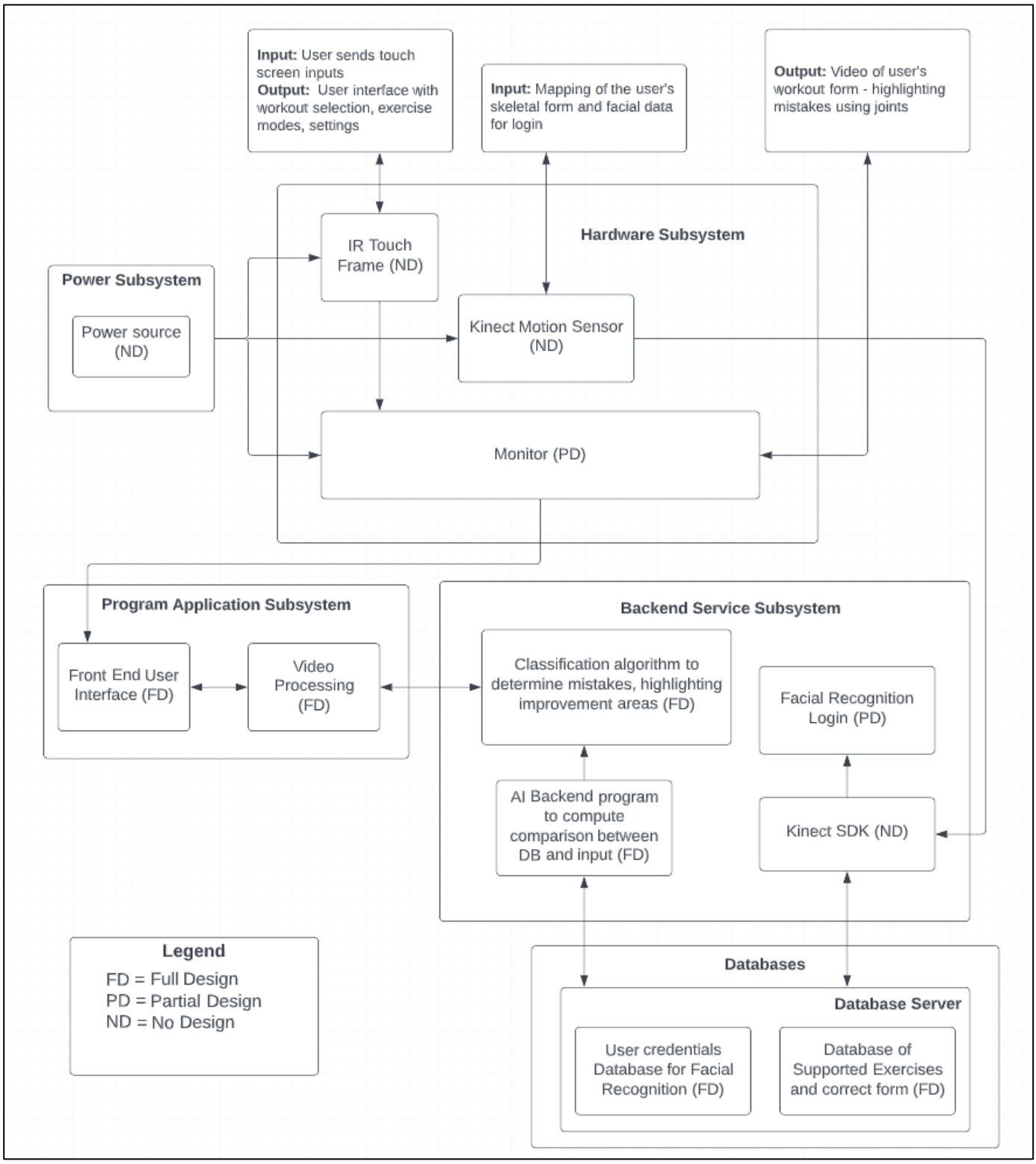


Figure 1: Smart Workout Trainer Subsystem Block Diagram

2. Project Specifications

Project specifications can be broken down between functional and non-functional specifications.

2.1 Functional Specifications

Table 1: Key Functional Specifications

No.	Subsystem	Specification	Details	Necessity
FS1	Software	Touch converted to user inputs	The software should allow for convert touch inputs to specify workout selection, exercise modes, settings	Non-Essential
FS2	Software	Sense user's form and body movements	The hardware should read and capture key features of the user's body movements including their form/posture in 3D. This includes depth, joints, and connections	Essential
FS3	Software	Display correct workout form/posture	The software should identify the correct posture at any given point in time of the exercise and prompt the user to correct their posture by overlaying the correct skeletal vectors on top of the user's current workout form.	Essential
FS4	Software and Backend Service	Authenticate different user accounts	The backend service should recognize and authenticate users to sign them onto their own personal and saved settings	Non-Essential
FS5	Software and Backend Service	Compare user's body form with correct form	For the prototype, the backend service contains a lookup table of correct skeletal vectors of all bones on the body at 30 frames per second. In order to compute the accuracy of the user's body posture, we compare the ideal skeletal vectors with the actual vectors real-time. If accuracy is less than 80%, meaning less than 80% of the skeletal vectors match, then the user should be shown corrected form overlaid on their original video	Essential
FS6	Program Application	Display video and user interface through an application	The program application should display an application showing user's movements with corrected movements. There should also be functional buttons, tools etc. to customize workout and modify settings	Essential
FS7	Database Server	Store user authentication data	The database should store user facial credentials to authenticate their account and personal settings	Non-Essential
FS8	Database Server	Store correct workout form	The database should store data regarding the supported exercises as well as the correct workout form/posture for the entirety of each exercise.	Essential

			We will do this by pre-loading ‘correct’ skeletal vectors at 30 frames per second. This is done by recording each exercise with the correct form and extracting the vectors using a computer program that reads and stores these vectors in our backend database.	

2.2 Non-functional Specifications

Table 2 outlines the non-functional specifications applicable for our design.

Table 2: Key Non-Functional Specifications

No.	Subsystem	Specification	Details	Necessity
NFS1	Hardware Sensor	Display Size	The screen size should be no smaller than 55’’	Non-Essential
NFS2	Hardware	Product Size	The entire product should not have a form factor larger than 30” x 50” x 3”	Non-Essential
NFS3	Hardware	Product Weight	The entire product should not weigh more than 22kg	Non-Essential
NSF4	Power	Power requirement	All the hardware components should receive continuous power. The power source should provide 250W to support the monitor and computational power required for the hardware components of the system	Essential
NSF5	Backend Service	Comparison Algorithm speed and robustness	The algorithm created for comparison must be fast enough and robust enough to accurately compare user input with correct form in real time. There should be a lag of no more than 5 fps (frames per second).	Essential

3. Detailed Design

3.1 Inputs

Table 3 decision matrix of input options

Criteria	Weight	Alternatives		
		Touch	Voice	Hand Gesture
Competency	/3	3	1	1
Usability	/7	7	5	6
Ease of development	/5	5	3	3
Total		15	9	10

Criteria Legend: when deciding the best way for accepting user inputs, many criteria were considered.

Competency:

> Competency refers to how capable the team is able to develop this option. 4 of 5 of our team members are comfortable in software development. Since touch input can be directly converted from mouse input, this method does not require additional learning. As none of the team members have experience in hand gesture and voice control APIs, it would require additional learning before developing.

Usability:

> Usability refers to how easy users can interact with the application and how error free their inputs can be. Touch input and hand gesture are both direct. As the user will have to learn the different hand gestures, it requires additional practice. Sometimes the camera will not pick up the correct gesture or mistakes it for another due to angle or speed, it will be more error prone.

Ease of development:

> East of development refers to how quickly the team can develop and if there are additional resources that are provided to simplify the process. As c# (the main development language) provides drag and drop templates when developing front end, touch input is quite simple to develop. Although voice control and hand gesture have variety of API and Kinect provides additional libraries for hand gesture recognition, it will require more research on top of regular development.

Overall decision: Touch Input

To avoid potential overhead, the team has decided to use touch input as the main user input method. As voice input and hand gesture are more user-friendly during workout sessions, they are kept as potential additional features the team can develop when proven capacity.

3.2 Outputs

Table 4 decision matrix of output options

Criteria	Weight	Alternatives	
		Visual	Voice
Competency	/3	3	1
Usability	/7	7	5
Ease of development	/5	5	3
Total		15	9

Criteria Legend:

Competency:

> Competency refers to how capable the team can develop this option. 4 of 5 of our team members are comfortable in software development. As visual feedback is what we have most experience on, it is the easiest to develop. Because voice feedback requires additional learning of the API and more logic to convert user joints' location to voice feedback.

Usability:

> Usability refers to how easy users can understand the feedback. Visual feedback is the easiest way users can understand their form's correctness. But it lacks accessibility for users who are visually impaired, for which voice feedback can be used.

Ease of development:

> East of development refers to how quickly the team can develop and if there are additional resources that are provided to simplify the process. Because visual feedback displays the correct skeletal form, it is very easy to implement. As voice feedback requires the additional logic of converting the vectored skeletal form into voice parameters, it would be more challenging.

Overall decision: Visual Feedback

The team has decided to use visual feedback as the main output method. Because voice feedback provides additional method for accessibility, it is kept as a potential feature for future development.

3.3 Hardware Subsystem

3.7.1 IR Touch Frame

Criteria	Weight	Alternatives		
		IR Touch Frame	Capacitive Screen	Optical Imaging Touch
Price	/5	5	1	3
Usability	/7	6	7	5
Ease of development	/5	5	5	5
Integration	/3	3	-	2
Total		19	13*	15

Criteria Legend:

Price:

> Price refers to the cost of component. A weight of 5 was chosen as the price is quite critical as it should not exceed project budget (600$) but also allow cost for other components. We found that the IR Touch Frame was cheaper than a capacitive touch screen and Optical Imaging touch.

Usability:

> Usability refers to the ease of use and reliability for user touch inputs. Capacitive touch screen has best precision and response time. The IR touch frame and Optical Imaging touch frames use IR and light optics to detect user touch points, as they rely on light/infrared technology they have

lower precision and response time. They can have “ghost” touches if a foreign object enters the touch zone.

Ease of development:

Ease of development refers to the additional development that would be required to use the different touch with our system. We found that all three system offers “plug and play” compatibility. i.e we would just plug in the device to our system and system OS would treat it as “normal” touch screen and handle all the inputs.

Integration:

Integration refers to the easy and capability of mounting the frame to our system in front of the mirror. The IR touch frame and optical imaging frame can be mounted on top of the mirror without interfering with its functionality, however the capacitive touch screen cannot be mounted ontop of the mirror and gets eliminated as a viable option.

Overall Decision:

The team decided to use the IR Touch frame due to its cost, integration and usability.

3.7.2 Kinect Sensor

Criteria	**Weight**	**Alternatives**		
		Kinect	**Depth Camera**	**Camera w/computer vision**
Price	/5	4	1	5
Ease of development	/7	7	5	4
Integration	/5	3	3	5
Total		14	9	14

Criteria Legend:

Ease of development:

Ease of development refers to the level and complexity of development the team would have to do to use the camera/sensor with the main program. For the camera option we would have to implement a computer vision system and use ML models to detect human skeleton points. Microsoft Kinect offers a well built and document api for us to directly extract human skeleton points.

Integration:

Integration refers to the special (if any) system requirements the camera/sensor needs. A camera doesn’t have any special system requirements. The Microsoft Kinect is plug and play via usb but has some requirements like USB 3.0 , 64bit processor and DirectX 11 support.

Overall Decision:

The team decided to use the Microsoft Kinect as it offers video and depth sensor with a very easy to use API for development.

Example Code for Kinect:

```
/// <summary>
/// Handles the body frame data arriving from the sensor
/// </summary>
private void Reader_FrameArrived(object sender, BodyFrameArrivedEventArgs e)
{
    bool dataReceived = false;
    using (BodyFrame bodyFrame = e.FrameReference.AcquireFrame())
    {
        if (bodyFrame != null)
        {
            if (this.bodies == null)
            {
                this.bodies = new Body[bodyFrame.BodyCount];
            }

            // The first time GetAndRefreshBodyData is called, Kinect will allocate each Body in the array.
            // As long as those body objects are not disposed and not set to null in the array,
            // those body objects will be re-used.
            bodyFrame.GetAndRefreshBodyData(this.bodies);
            dataReceived = true;
        }
    }
    if (dataReceived)
    {
        foreach (Body body in this.bodies)
        {
            if (body.IsTracked)
            {
                IReadOnlyDictionary<JointType, Joint> joints = FilterJoints(body.Joints);

                // convert the joint points to depth (display) space
                Dictionary<JointType, Point> jointPoints = new Dictionary<JointType, Point>();
                foreach (JointType jointType in joints.Keys)
                {
                    // sometimes the depth(Z) of an inferred joint may show as negative
                    // clamp down to 0.1f (-Infinity, -Infinity)
                    CameraSpacePoint position = joints[jointType].Position;
                    if (position.Z < 0)
                    {
                        position.Z = InferredZPositionClamp;
                    }

                    DepthSpacePoint depthSpacePoint =
                                                coordinateMapper.MapCameraPointToDepthSpace(position);
                    jointPoints[jointType] = new Point(depthSpacePoint.X, depthSpacePoint.Y);
                }
                this.DrawBody(joints, jointPoints, dc, drawPen);
                this.DrawHand(body.HandLeftState, jointPoints[JointType.HandLeft], dc);
                this.DrawHand(body.HandRightState, jointPoints[JointType.HandRight], dc);
            }
        }
    }
}
```

3.7.3 Monitor

The team decided to use a 55” TV as this seems a good size to be clearly able to see the visual feedback and also fits within our budget. The screen size also influences the mirror size and IR touch frame size (bigger the size, the more expense each piece gets). Hence we were able to find the different pieces: two way mirror, screen, IR touch frame of 55” size and within our budget.

3.7.3 Hardware Assembly

As wall mounting would not have been a suitable option for the symposium, we had decided to mount the entire system on a portable tv cart (as seen below). This not only gave us a mounting option but also a portable system which we could move around and also adjust the height of the system as needed.

The screen was mounted onto the tv cart using a standard vesa mount, the two way mirror is placed ontop of the screen using standard j-hooks and adhesive, the IR touch frame is mounted on top of the mirror using simple adhesive. A simple enclosure is made to house the Kinect Camera either on top or below the monitor. Refer to follow images for dimensions, visual renders.

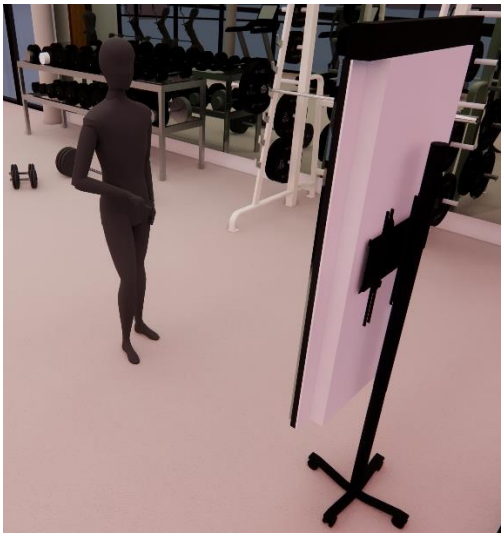
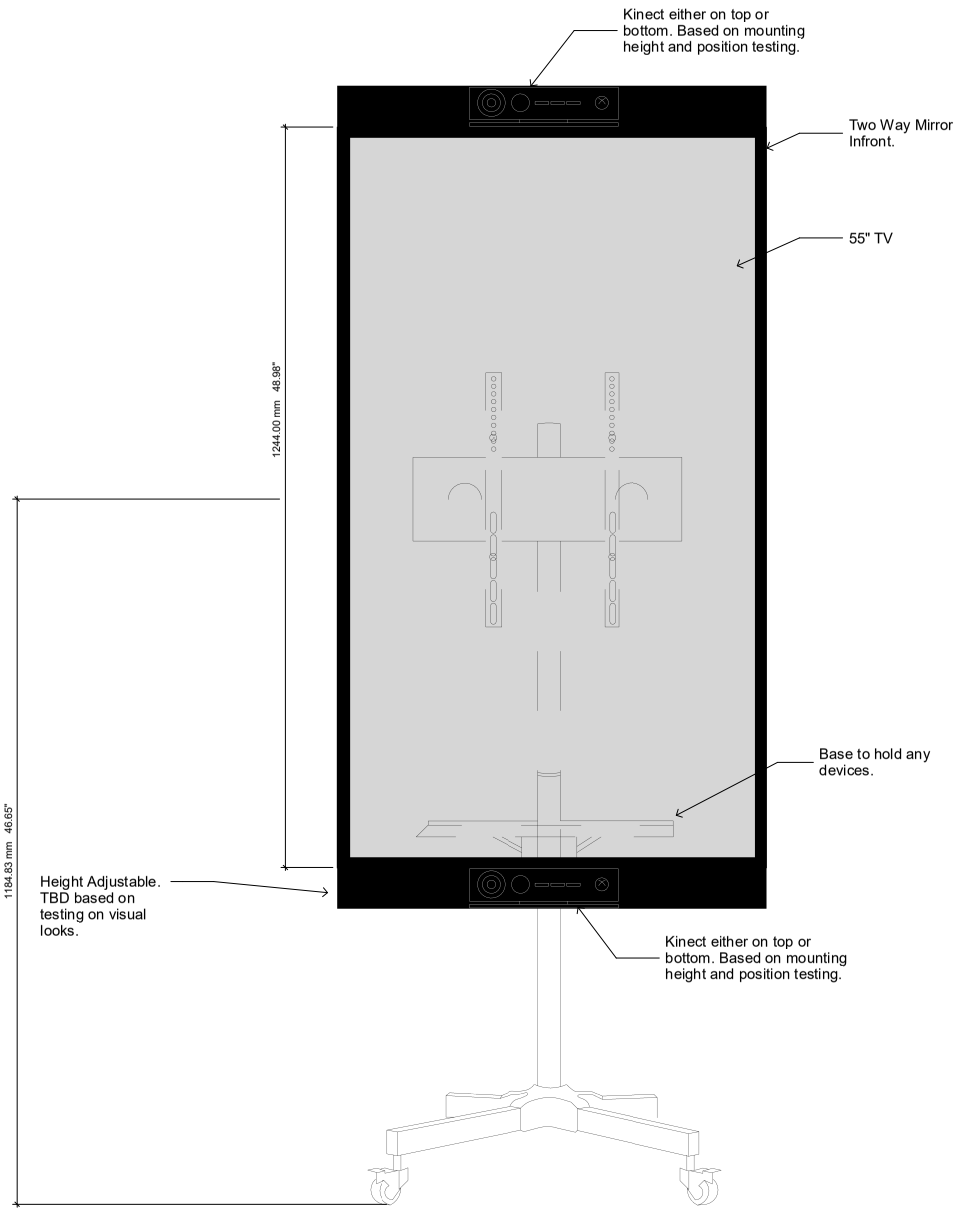
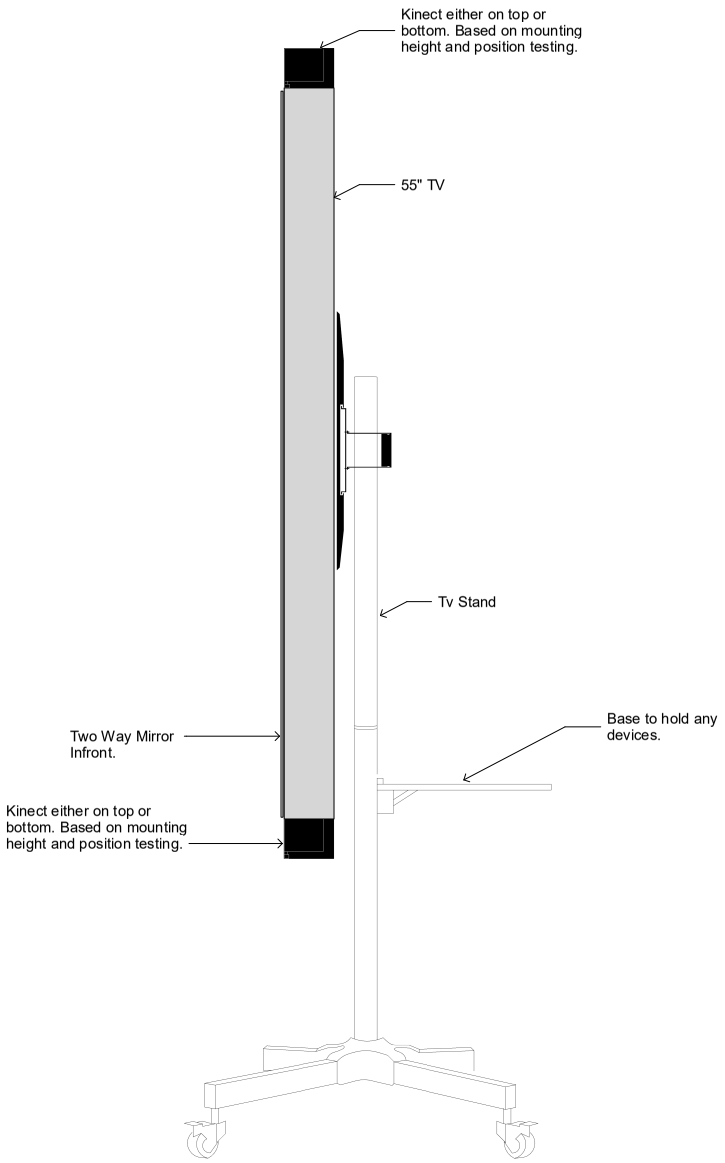


Figure 2,3,4 & 5: Display showing the hardware assembly

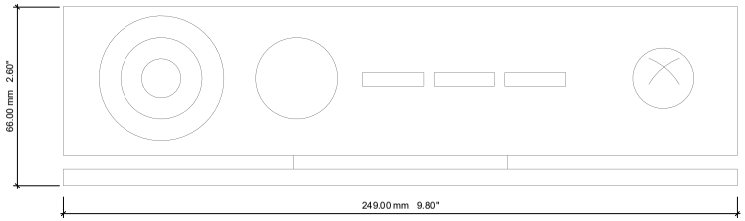
FRONT VIEW.



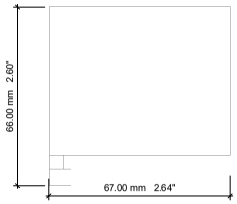
SIDE VIEW.



FRONT VIEW



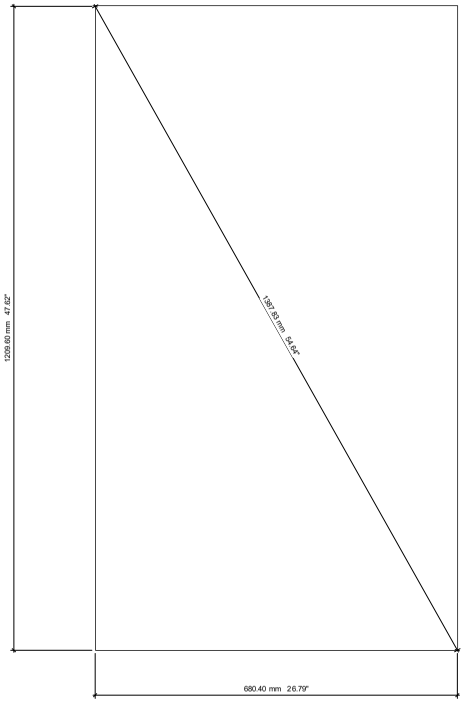
SIDE VIEW



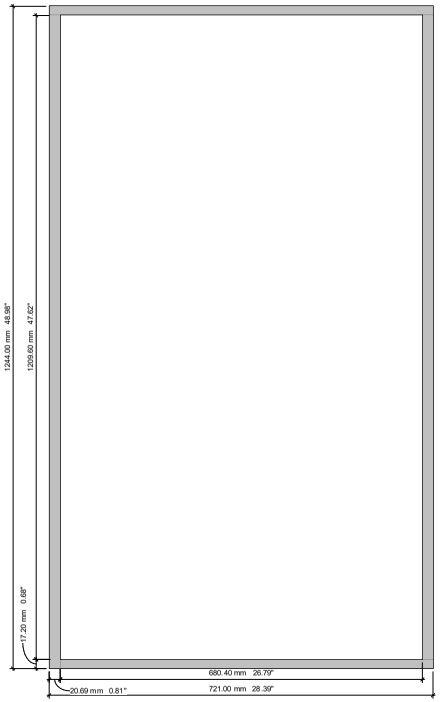
POWER CONSUMPTION:

- Voltage: 12V
- Current: 2.67A
- Max Power Consumption: 32.04W
- Average Power Consumption: 16W/hr

SCREEN DIMENSIONS



SCREEN WITH BORDER DIMENSIONS



SIDE VIEW

POWER CONSUMPTION:

- Voltage: 120V
- Current: 1.4167A
- Max Power Consumption: 170W
- Average Power Consumption: 80W/hr

3.4 AI Subsystem

The AI subsystem is the technical portion responsible for determining improper workout form and highlighting areas of improvement. As such it requires us to create an implementation that analyzes a user's movements against a set of expected/correct movements. Our team came up with three possible solutions to the AI implementation, as well as two different neural networks considered for the machine learning approach.

3.3.1 Lookup Table

The first implementation uses the idea of a lookup table, which we would use to index the proper and wrong poses for the cycle of a movement. This method would require us to build a lookup table. The overall logic would flow in this manner: the process first detects which part of the exercise the user is in, then the program indexes the lookup table for correct poses and compares it with the exercise data in the database. The advantage of this design is in its simplicity, but it is burdened by its high latency, as the program will need to perform search processes.

3.3.2 Machine Learning with Multiple Labels

Alternatively, a more complex and reliable approach would be to train a ML model. By having a person perform the correct movements of an exercise, then having a training model sample all the joint positions and angles throughout the movement we can effectively train a LSTM/CNN model to classify whether a user's form is performed correctly or incorrectly. To develop the model, we would encode the "correct" exercise examples to efficiently use them as training data for the model. By leveraging a LSTM/CNN model, we can boast a reliable speed for classifying the correctness of workouts, as the model was trained offline.

As mentioned, we considered the convolutional neural network (CNN) and the Long short-term memory approach to creating our machine learning implementation.

3.3.2.1 Convolutional Neural Network

The Convolutional Neural Network (CNN) is a deep learning model commonly applied to image recognition tasks.[1] However, for our design we must analyze the entire duration of an exercise, which means we must deal with video analysis as opposed to images. Analyzing video data is considerably more difficult than images as we need to consider an extra temporal dimension. While there are approaches to applying a CNN for video analysis, such as fusing two CNNs to deal with both the spatial and temporal streams, there exist other methods to solving this problem[2]

3.3.2.2 Long short-term memory

The long short-term memory (LSTM) is an artificial neural network. The advantage of the LSTM is that it is a recurrent neural network, meaning it can process not just a single point of data but also entire data sequences.[3] The LSTM uses the output of previous results as inputs to the current step. This property of the LSTM makes it desirable for our workout trainer, as when analyzing a video of the user's form, every

[1] Valueva, M. V., Nagornov, N. N., Lyakhov, P. A., Valuev, G. V., & Chervyakov, N. I. (2020, May 6). Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. Mathematics and Computers in Simulation. Retrieved June 21, 2022, from https://www.sciencedirect.com/science/article/abs/pii/S0378475420301580?via%3Dihub

[2] Large-scale video classification with Convolutional Neural Networks. (n.d.). Retrieved June 22, 2022, from https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Karpathy_Large-scale_Video_Classification_2014_CVPR_paper.pdf

[3] Google research - static.googleusercontent.com. (n.d.). Retrieved June 22, 2022, from https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43793.pdf

movement the user makes will have been a result of a previous movement to make it to that point.[4] By using an LSTM neural network, we can accurately classify the integrity of a user's workout form.

3.3.3 Degree of Correctness

The final implementation we considered was degree of correctness. For any given exercise, we will have multiple examples of a correct interpretation of that exercise. The degree of correctness idea has us search through all the correct examples of the current work out and find how the current user's movement is deviating against the correct examples. We accomplish this by searching for the closest correct match to the user's input, then we look for the margin in which the user deviates from the correct example by using regression. In this way, we can determine the rate of error in which a user is within a correct example and analyze if the rate of error is small enough to be considered correct.

Criteria	Weight	Alternatives		
		Lookup Table	Machine Learning (LSTM)	Degree of Correctness
Accuracy	/6	4	6	5
Ease of development	/3	3	1	2
Speed & latency	/5	3	5	4
Total	14	10	12	11

Overall decision: Machine Learning (LSTM)

Thus, to create an effective AI system that has a high degree of accuracy and minimal latency, we will implement the machine learning approach using long short-term memory to classify the correctness of a user's workout form.

3.5 Program Application Subsystem

3.4.1 Software Options

Table 5 decision matrix of software options

Criteria	Weight	Alternatives	
		Webapp	Native
Competency	/3	3	2
Security	/5	2	5
Speed & latency	/7	4	6
Ease of development	/5	3	5
Flexibility	/5	5	3
Total		17	21

Criteria Legend:

Competency:

> Competency refers to how capable the team is able to develop this option. 4 of 5 of our team members are comfortable in web development, while 1 member is fully comfortable in developing in C#, which would be the main language used in the Native app approach. However,

[4] Towards a digital personal trainer for health clubs - researchgate. (n.d.). Retrieved June 22, 2022, from https://www.researchgate.net/publication/322879733_Towards_a_Digital_Personal_Trainer_for_Health_Clubs_-_Sport_Exercise_Recognition_Using_Personalized_Models_and_Deep_Learning

given our experiences in learning new languages and the relatively low learning curve for C#, developing for the native app should not pose as too difficult in comparison to the web app.

Security:

Security refers to the ability to keep user data secure. As webapp requires encryption and decryption when sending user data to and from servers, it is more prone to security flaws. Since native does not require to be connected to the internet during user sessions, it is less prone to security attacks and data leakage. Finally, both methods require encryption for user login to ensure secure user access.

Speed & Latency:

Speed and latency refer to the app's ability to show feedback to the user in real time. The Native app offers better latency in this regard because it will run as a local executable file on the smart mirror interface. As providing real-time feedback is one of the core features of this product, this is an incredibly important factor to consider and is therefore given a higher weight. The webapp route would require connecting to a webserver and running our AI backend via cloud services, which could introduce higher latency.

Ease of development:

East of development refers to how quickly the team can produce a working product. Although both webapp and native will have lots of documentation and online support. As webapp requires more subsystems (front end, backend, middleware), it is more complex.

Flexibility:

In terms of creating a flexible app that can be turned into a commercial product, the webapp approach would allow for greater scalability as one server can handle multiple user requests and software updates are easier when all applications are connected to the same server.

Overall decision: Native app

To avoid potential overhead in using a webapp, for our purposes, we will be developing a C# based native app. This way, we can focus our attention on building an accurate and consistent real-time workout posture correction AI model to provide the greatest benefit for the user.

3.4.2 Visual Feedback

The following images are visually what the native application looks like. It will have color-coded skeletons displaying user's form. Based on correctness, additional color-coded skeletons will appear on screen displaying the correct posture.

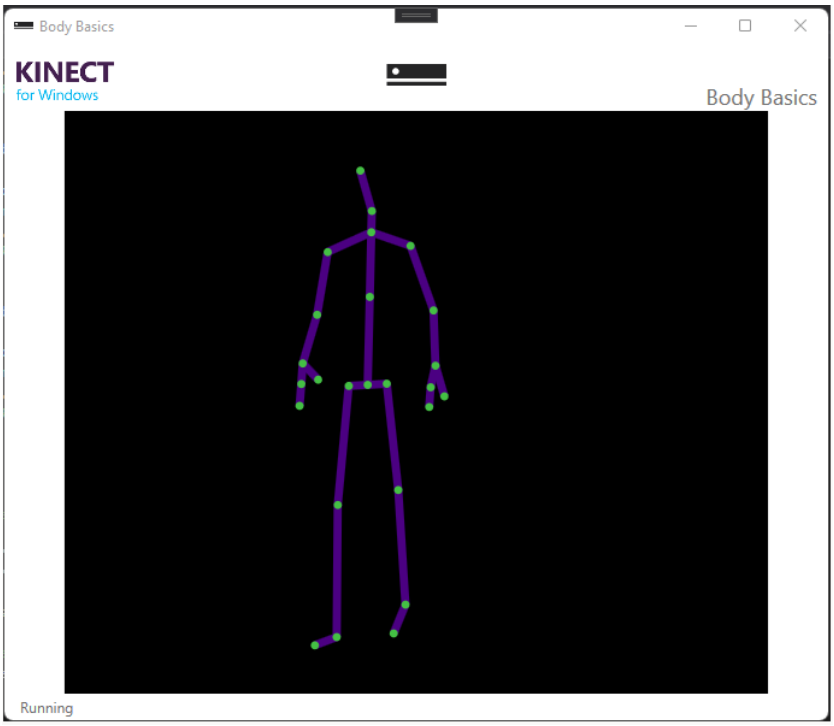


Figure 6 Skeletal display in 2D view

While the above image displays a skeleton in 2D, there are options to display it in 3D (figure 3 and 4).

This gives user a better perception in depth, allowing easier error correction for the user.

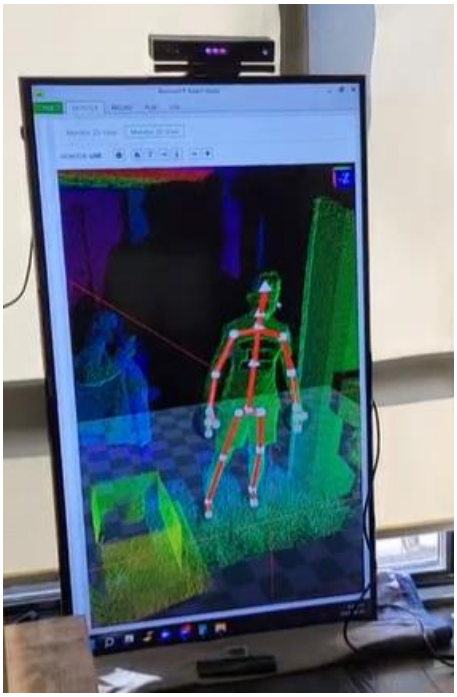


Figure 7 & 8 Skeletal display in 3D view, Skeletal display in 3D view showing depth

3.4.3 User Session

Table 6 decision matrix of user session options

Criteria	Weight	Alternatives	
		Temporary	**Permanent**
Ease of development	/5	5	3
Security	/5	5	3
Feature Expansion	/7	3	7
Personalization	/5	1	7
Total		14	20

Criteria Legend:

Ease of development:

> East of development refers to how quickly the team can develop this functionality. As the temporary user session does not require additional database generation, it does not require any additional development. Whereas for permanent user session where each user is required to generate user profile and login information.

Security:

> Because temporary user session does not take in any user information, it does not possess any security threats. The permanent user session will require more security measures to protect user data.

Feature Expansion:

> Feature expansion refers to the ability the application can branch additional features based off of current design. When user session is temporary, lots of useful information are not stored, thus limited features can be branched off of current design; whereas when each user have a user profile, all the user information collected can be used to personalize user workout, better guide their sessions and provide useful history to their workout.

Personalization:

> Personalization refers to the ability the application can store personalized information about each user. Temporary user session does not store any information whereas permanent user session supports customization such as customized user workout routine, favorite workout sets and customized break time.

Overall decision: Permanent User Session

The team decided to use permanent user sessions as it provides more freedom in feature development.

3.6 Backend Service Subsystem

3.5.1 Authentication

Using the Kinect sensor, we employ a facial recognition authentication service that allows the user to login to use the program. In this manner, we can identify the user and load the data they share with the AI trainer. The user credentials and facial recognition data will be stored in a local database.

3.5.2 Classification Algorithm – refer to AI Subsystem (Section 3.3)

The primary component of the backend system is the AI component that classifies the degree to which a user's workout form is correct. The classification algorithm, using AI, will record the user's workout form and compare it against "correct" examples that were used to train the model offline. The specific AI technique we have chosen is using a long short-term memory recurrent neural network. For more information, refer to section 3.3 that outlines the different solutions and AI approaches our group considered.

3.5.3 Program to load data of correct skeletal points from database

In our comparison between real-time user movements and correct skeletal points (when using the 'degrees of correctness' approach) there needs to exist frictionless way to search for a) the point at which a user is in a given exercise (ie: the lowest point of a pushup) and b) the 'degrees of correctness' by which the user is off by given vector calculations of the user's joints in space. To efficiently make these comparisons, there needs to exist an easy way for information about correct workout movements to be quickly searched up. In this way, we need to consider how to store x, y, z information into a database. The simplest method would involve storing these tuples or objects (x, y, z) into the DB directly, but this could be slow for search potentially. In order to efficiently search through our database, we can use a heuristic search method to predetermine which rows or columns could be relevant to our needs at any given time.

3.7 Database Subsystem

For the purposes of the AI workout trainer being built, our team entertained the possibilities of using a relational and non-relational database. From a high-level perspective, the primary use case for our database was to store "correct" workout examples for all our supported exercises. To store a "correct" workout example, we use an encoded data format that provides information on all the joint locations, angles, and joint type for the duration of an entire exercise and insert that information into the database.

To create a relational database with these data fields, we considered creating tables for the exercise, joint type, and joint locations and then with these tables we could create relations between them. However, this would result in data duplication, and the entity relationships would be unintuitive when creating queries. This would lead us to consider using a non-relational database for our design.

With a non-relational database, our database will have large amounts of data with simple and predictable query patterns. Therefore, we decided to use a non-relational database as they provide fast queries and flexible data models.

The schema for our database can be seen as follows:

Data Field	**Type**	**Description**
exercise_type	String	The type of exercise being evaluated
muscles_worked	List of strings	The muscle groups worked by the exercise
correct_example	Object	An object that stores the correct movements performed throughout an exercise. This object stores the joint_data field as well as the frame.
frame	Int	The frame represents a unit of time that keeps track of which step the correct exercise example is currently on.
joint_data	List of objects	The joint_data field keeps track of the joint, position, and depth data at the current frame
joint_type	String	joint_type is the name of the joint currently being analyzed
position_point	List of doubles	Position point is an array of three doubles, it illustrates the current location of the joint's coordinates (x, y, z)

depth_space_point	List of doubles	The depth space point illustrates the 3D depth of the joint (x, y)

Thus, an example document would look like the following:

```
{
    "exercise": "Bicep Curl",
    "muscles_worked": ["bicep", "brachialis", "brachioradialis"],
    "correct_example": [
        {
            "frame": 0,
            "joint_data": [ {
                "joint_type": "SpineShoulder",
                "position_point": [0.05033442, 0.214498, 1.768061],
                "depth_space_point": [271.4493, 164.3408]
            },
            {
                "joint_type": "HandTipLeft",
                "position_point": [-0.1455134, -0.3222967, 1.885995],
                "depth_space_point": [232.7272, 271.4432]
            },

            "joint_type": ...
            ]
        },
      {
 // This repeats for the # of frames recorded
          "frame": 1...
                  ...
                  ...
          "frame": n
      }
    ]
}
```

In the above example, we can see that for any exercise there will be a lot of data stored as we keep track of the joint_data for all the frames analyzed in an exercise, meaning that the size of the correct_example list is equivalent to the number of frames we analyze. Moreover, the size of the joint_data list is equivalent to the number of joints we analyze (25).

4. Discussion and Project Timeline

4.1 Evaluation of Final Design

The main objective of our project is to design a product that can monitor and correct posture in real-time for people who are working out in order to reduce the risk of injury from overexertion. Further, we aim to accommodate different body types and meet the accessibility needs of diverse user groups by building a product that is inclusive and accessible. We also want the solution to be affordable and scalable compared to close alternatives and track key metrics overtime.

The proposed design in this report meets this objective at a high level. Below, the table breaks down some of the key objectives and how they are met in our proposed design.

Table 5: Criteria for meeting objectives

Objective	**How is it met?**
Monitor and correct posture in real-time	AI component (3 options)
Accommodate different body types	Using scaling for "bone vectors" regardless of body-type in order to generate a constant pre-image before computation
Build a product that is accessible	Using a technological solution, we can ensure that people can workout at any time and place
Build a solution that is affordable	Our prototype is built with <$600 which is more cost-effective then industry Smart Workout Smart Mirror solutions that range from $1000 - $5,000 [10]

However, further considerations need to be made regarding how to meet accessibility needs and how key metrics will be tracked. For accessibility, voice activation/feedback is a possibility, further consideration is being made about the prioritization of this component given the limited time available.

As well, we are still brainstorming what key data points need to be tracked over time (I.e., workout duration, workout score based on degree of correctness of aggregated workout) and how it can be integrated into the design of the solution.

4.2 Use of Advanced Knowledge

This project uses upper year ECE knowledge of three main subject areas: ECE 356 Databases, ECE 358 Networking, ECE 457 A Cooperative and Adaptive Algorithms and ECE 458 Computer Security.

Databases are used for designing the backend subsystem to store and retrieve data about correct body postures. For this, we will use ECE 356 knowledge in order to explore the database system architecture and database design in order to effectively achieve fast retrieval of data to correct body posture in real-time

Also, as we are working to develop a Native App, it will require our AI component to use .NET Framework in order to send messages over the network. In this sense, we will be exploring architecture styles, particularly on top of the application layer that can help send data.

Further, some possible solutions relating to the AI component will require using a look-up table in order to determine which part of the exercise the user is to offer correction. For this, we are exploring local search methods involving better heuristics to search and retrieve a solution from the database faster.

Finally, ECE 458 knowledge on entity authentication will become a useful application to consider particularly as we consider a user-login system.

[10] Halse, K. (2022, February 2). *6 best smart fitness mirrors for home workouts*. Heavy.com. Retrieved June 27, 2022, from https://heavy.com/sports/smart-fitness-mirror/

4.3 Creativity, Novelty, Elegance

This project is novel because our proposed design is much cheaper (costing <$ 600 to make) and in some cases more scalable compared to existing alternatives in the long run. For instance, personal trainers are

expensive, costing between $30 and $100/hour. With the help of our project, we help train and improve workout performance while offering the ability to scale more effectively compared to trainers who are constrained by expertise, location, and hours of flexibility.

Also, even compared to existing Smart Mirrors that offer posture correction costs anywhere $1000 - $5,000, our solution is much more cost-effective per unit of production. Further, in-app solutions usable on a mobile phone device require a long-term subscription from the user, which far supersedes the cost of our Smart Mirror solution (which is a one-time cost), in the long term.

As we continue to develop our prototype, we plan to get more creative in the way we integrate features to meet greater accessibility needs of our users. In this way, we would like to meet the needs of "extreme user groups" in the market – a group that our competitors are not necessarily targeting at this level of their innovation.

One example of elegance can be seen in the way we strategically chose to use existing systems such as the Kinect Sensor which come equipped with built-in applications for depth and 2-D visualization to focus on creating a more robust AI-subsystem and user-interface.

4.4 Student Hours

The total number of student hours spent May 3rd – June 21st 2022 for each student are:

Table 6: Student hours logged

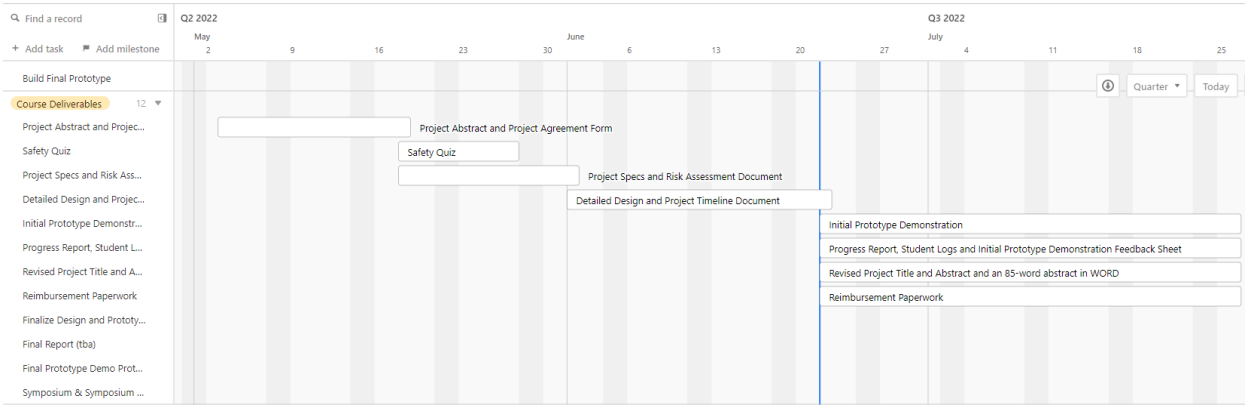
Student Name	**Total Hours Logged**
Rupakeerthana Vemulapalli	89
Rajan Sood	97
Wen Hao Li	91
Lucy Han	93
Gerry Zhang	92

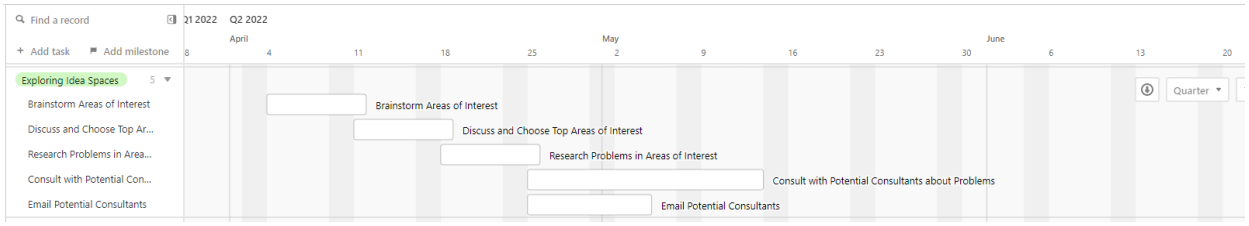
4.5 Potential Safety Hazards

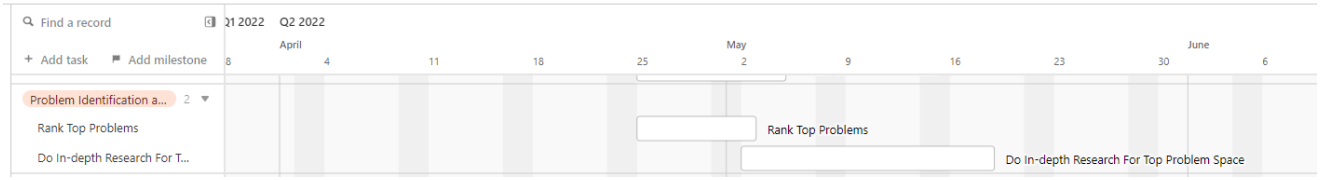
Since a large component of our project is purely software, there are no major physical safety hazards.

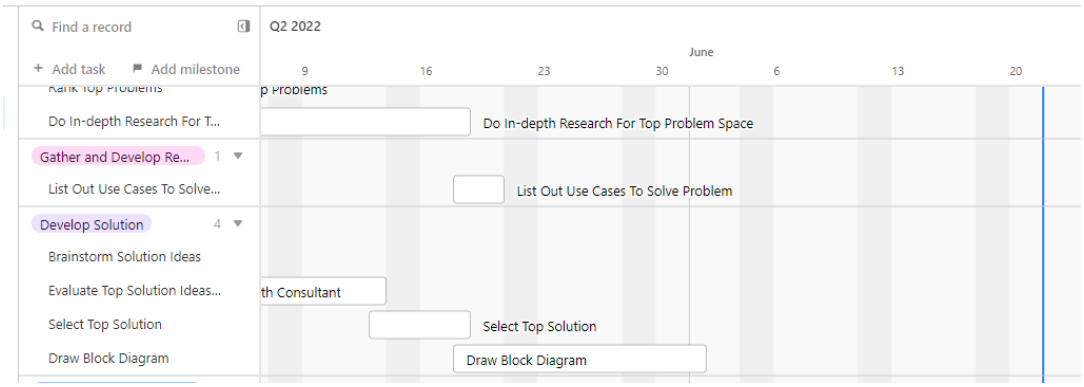
One potential area of threat could be from the Power subsystem component. Since the Kinect Sensor box needs to be plugged into a power outlet in order to operate, we must ensure that there are no exposed wires – this has the potential to cause an electric shock.

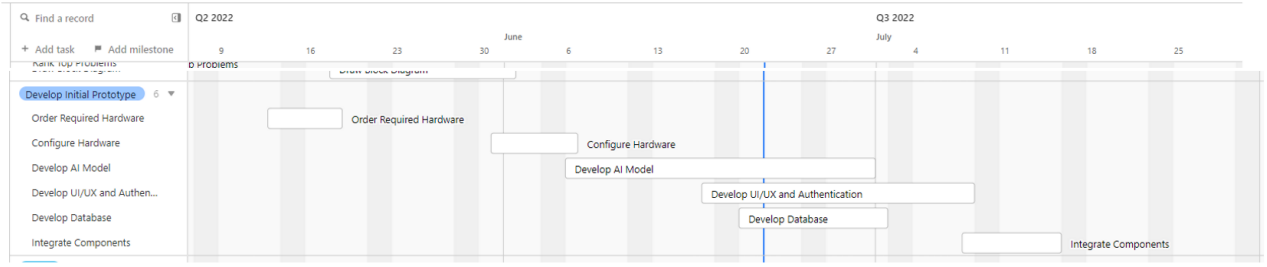
4.6 Project Timeline (Gantt Chart)

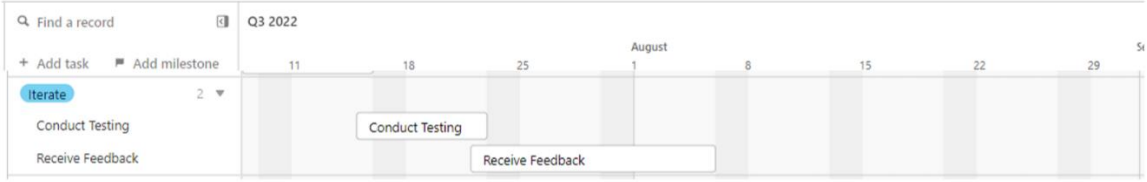












5. References

[1] At-Home Exercise Injuries Skyrocket During COVID-19 Pandemic | NEISS Data Study. Retrieved May 30, 2022 from, https://www.medicareadvantage.com/news/covid-at-home-exercise-injuries-report.

[2] Gray, S. E., & Finch, C. F. (2015, December). The causes of injuries sustained at fitness facilities presenting to victorian emergency departments - identifying the main culprits. Injury epidemiology. Retrieved May 16, 2022, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5005555/

[3] Nunez, K. (2020, December 8). Overexertion definition, signs and symptoms, prevention. Healthline. Retrieved May 16, 2022, from https://www.healthline.com/health/overexertion#symptoms

[4] Valueva, M. V., Nagornov, N. N., Lyakhov, P. A., Valuev, G. V., & Chervyakov, N. I. (2020, May 6). Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. Mathematics and Computers in Simulation. Retrieved June 21, 2022, from https://www.sciencedirect.com/science/article/abs/pii/S0378475420301580?via%3Dihub

[5] Large-scale video classification with Convolutional Neural Networks. (n.d.). Retrieved June 22, 2022, from https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Karpathy_Large-scale_Video_Classification_2014_CVPR_paper.pdf

[6] Google research - static.googleusercontent.com. (n.d.). Retrieved June 22, 2022, from https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43793.pdf

[7] Towards a digital personal trainer for health clubs - researchgate. (n.d.). Retrieved June 22, 2022, from https://www.researchgate.net/publication/322879733_Towards_a_Digital_Personal_Trainer_for_Health_Clubs_-_Sport_Exercise_Recognition_Using_Personalized_Models_and_Deep_Learning

[8] “In Which OSI Layer Is the Rest API Paradigm?” Stack Overflow, 1 Oct. 1962, https://stackoverflow.com/questions/29264855/in-which-osi-layer-is-the-rest-api-paradigm.

[9] “Electrical hazards”. Retrieved June 17, 2022 https://www.ccohs.ca/teach_tools/phys_hazards/electrical.html#:~:text=A%20plug%20or%20electrical%20cord,of%20electric%20shock%20or%20electrocution

[10] Halse, K. (2022, February 2). *6 best smart fitness mirrors for home workouts*. Heavy.com. Retrieved June 27, 2022, from https://heavy.com/sports/smart-fitness-mirror/