Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

ExerSights: AI Coach/PT

Submitted to

Osama Alshaykh
8 St. Mary's St Boston, MA 02215
Photonics Building Room 430
(617) 353-0223
osama@bu.edu

by

Team 2
Team ExerSights

Team Members

James Knee jknee@bu.edu
Anish Sinha anishs@bu.edu
Howell Xia howellx@bu.edu
Jilin Zheng jilin@bu.edu

Submitted: April 17, 2025

ExerSights User Manual

Table of Contents

Executive Summary – James

Fitness coaches and physical therapists (PTs) often teach their clients a set of exercises as part of a fitness/recovery routine. However, when clients perform these exercises on their own time, their coaches or PTs are not present to verify whether they are maintaining correct form.

Furthermore, many people lack easy access to coaches or PTs, so their form may never be assessed at all. Improper form for exercises can lead to long-term injuries and/or slow down personal fitness progress.

Our solution to this common problem is ExerSights. ExerSights is an AI Fitness Coach/PT application that leverages computer vision models to judge and correct a user's form on particular exercises and movements. Users are automatically provided exercise feedback from the application, enabling them to perform exercises safely and effectively on their own.

As part of the final deliverable, users can either upload existing videos or activate their device webcam to receive real-time text and audio feedback on a catalog of exercises. This feedback allows the user to immediately correct improper form, reducing the likelihood of injury and improving the efficacy of their workout.

1 Introduction - James

With increasingly sedentary lifestyles and an aging population, physical therapy has the potential to positively impact more people than ever. A March 2024 poll revealed that 51 percent of participants had received care from a physical therapist, and among those, 86 percent found it beneficial. Despite its growing importance, physical therapy faces challenges in ensuring patients perform exercises correctly. Treatment typically involves exercises requiring proper form to be effective. While physical therapists can guide and correct patients during in-person sessions, it is impractical for a therapist to be present for every session. This shortcoming creates a risk of patients performing exercises incorrectly, which can hinder recovery, reduce benefits, or even cause further injury. This issue also extends to personal fitness. Resistance training, a cornerstone of many fitness routines, shares similar risks when performed with incorrect form. While personal trainers can help address these issues, they may not always be affordable, accessible, or available. Consequently, there is a growing need for automated or remote solutions to help individuals maintain correct form during both therapeutic and fitness exercises.

Our project addresses this gap by developing a computer vision-based software solution that provides users with real-time feedback on exercise form, ensuring safety and effectiveness for individuals practicing physical therapy or personal fitness independently. Users will be able to tell if their form doesn’t match the exercise specifications and make corrections instead of having to guess if they are doing the exercises correctly.

Our software leverages computer vision to track user movements in real-time and compare their form against predefined exercise specifications. It then delivers real-time visual and audio feedback to guide corrections and correct improper form, while counting repetitions automatically. Our solution provides users with the option to livestream through the device webcam or to upload existing videos. The website is accessible 24/7, making guidance available whenever users need it. Users can choose from an extensive catalog of exercises and can also personalize each exercise’s settings (such as target depth for squats). The codebase is open-source, ensuring transparency with our solutions, feedback, and user privacy. The app is compatible with all devices(laptop, mobile, and tablet), so users can access it both on the go and at home.

By addressing the challenges of accessibility and proper form with real-time feedback, our project empowers users to safely and effectively perform physical therapy and fitness exercises at their convenience. This solution not only reduces the risk of injury but also broadens access to quality exercise guidance, fostering better health outcomes for all.

2 System Overview and Installation - Anish

2.1 Overview block diagram

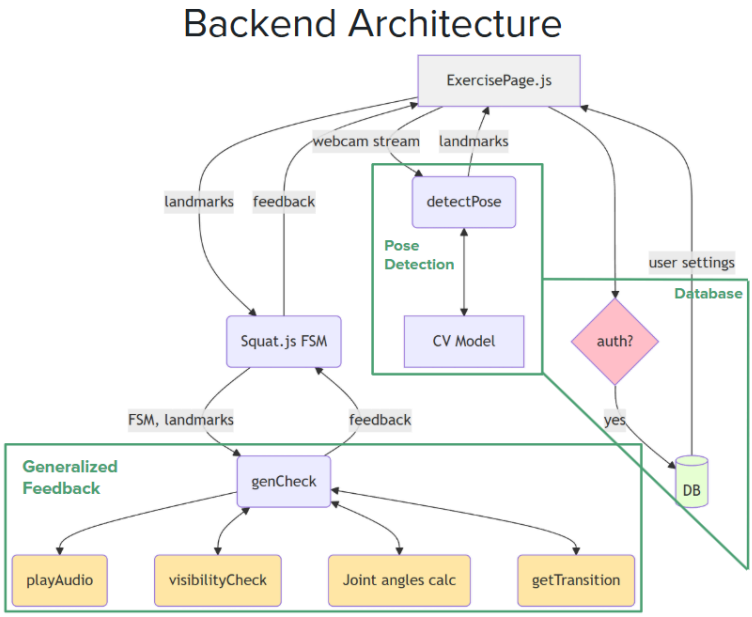

Figure 2.1.1 Each exercise page pulls the body joint positions from the MediaPipe computer vision model via our detectPose API. This data gets passed to the exercise finite state machine logic, which uses the genCheck method to perform various calculations and retrieve user feedback. This feedback is then passed back to the front exercise page. User-specific exercise settings are pulled from our database if the user is authenticated.

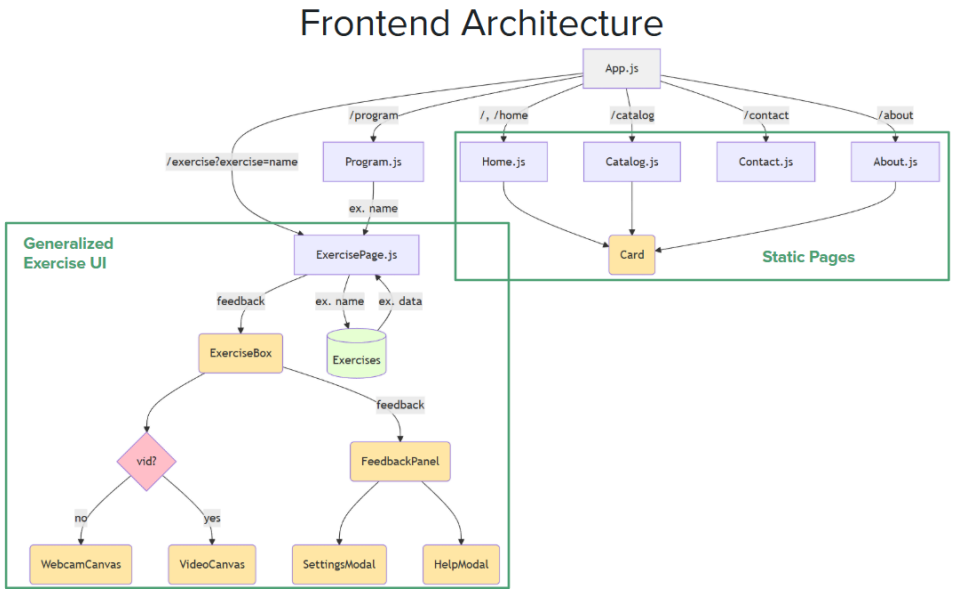

Figure 2.1.2 The App.js file serves all of the website routes. Each page pulls various front-end components. The exercise page pulls the ExerciseBox component, which populates the page with the feedback panel, settings modal, etc.

2.2 User interface.

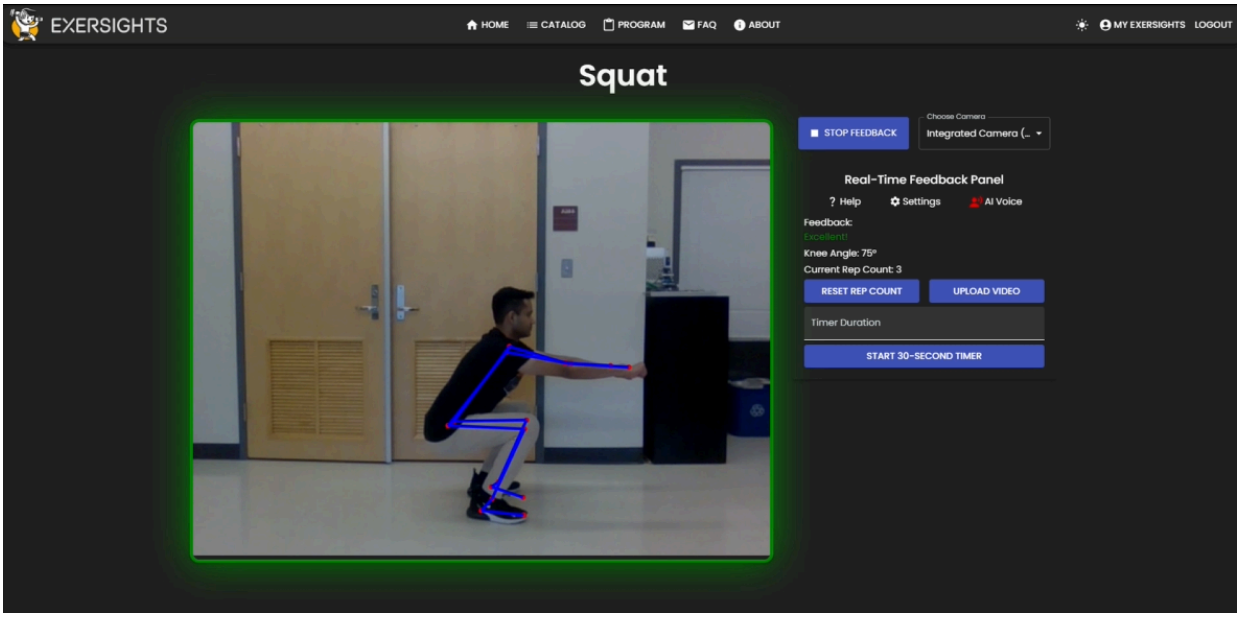

Figure 2.2.1 The exercise page overlays visual feedback onto the user's webcam and shows feedback on the right. The feedback panel has buttons to open the settings/help modals. There are also buttons to reset rep count, upload videos, and start timers.

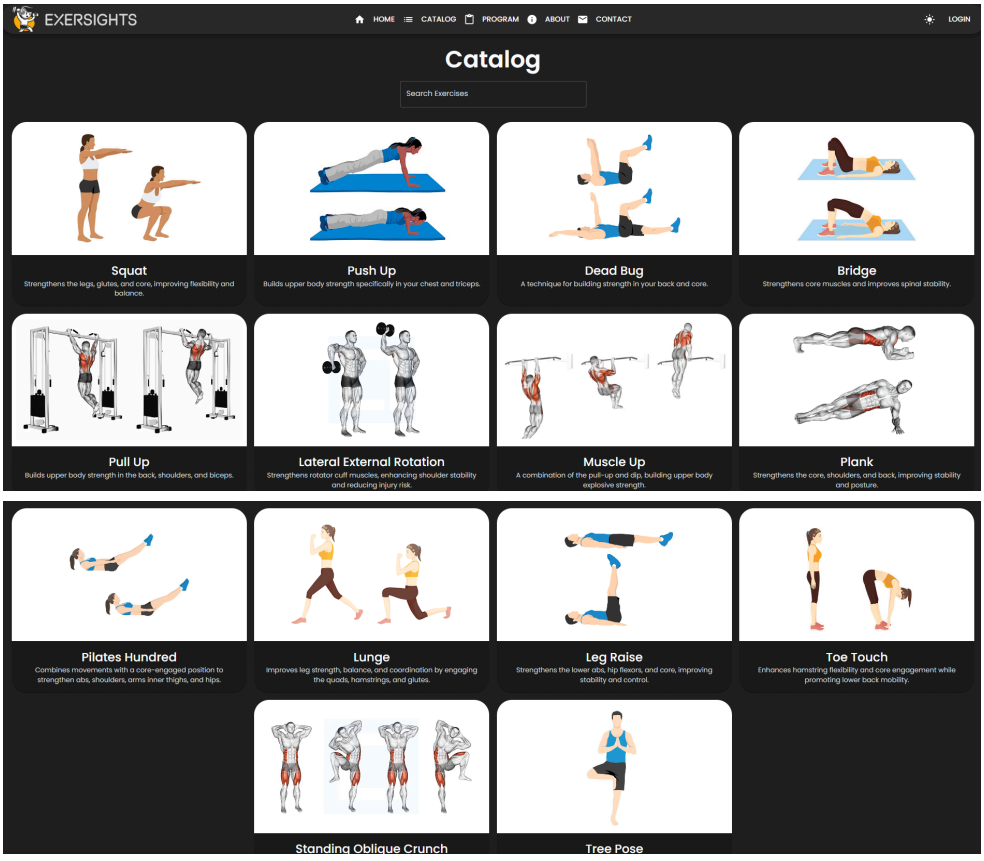

Figure 2.2.2 The catalog page lists all of the supported exercises and has a search bar.

2.3 Physical description.

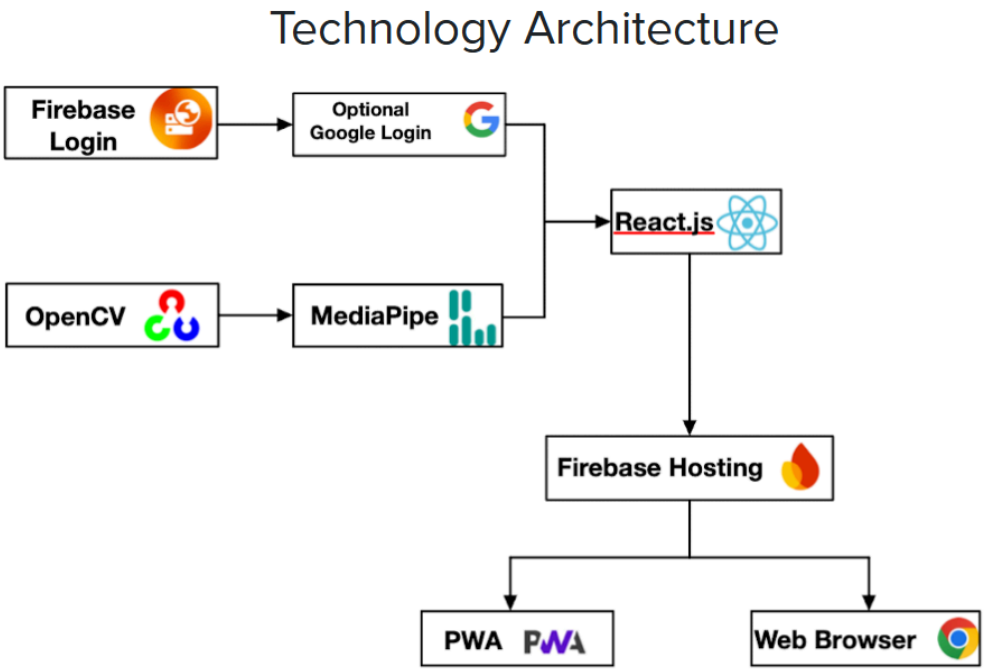

Figure 2.3.1 ExerSights uses MediaPipe and OpenCV computer vision libraries to handle human body detection. User authentication is handled by Google/Firebase. The site is hosted on Firebase and served both on the browser and as a progressive web app. The application is written using the React.js library.

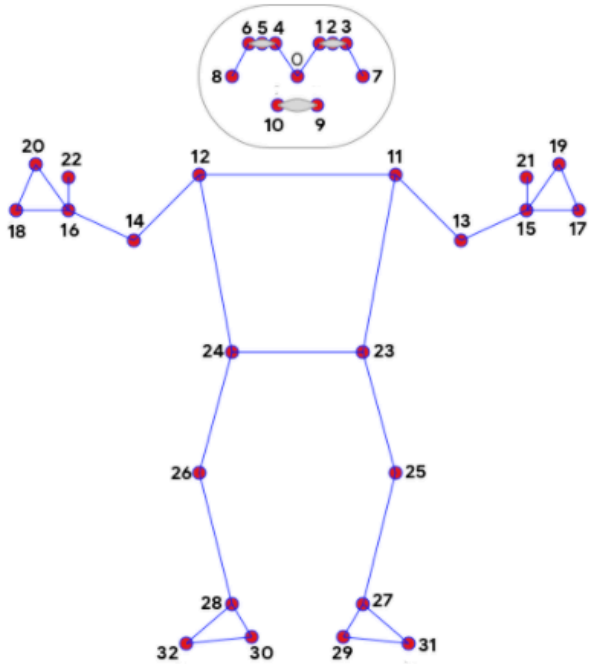

Figure 2.3.2 The MediaPipe computer vision library detects and labels 32 joints of the human body by processing images or video frames.

2.4 Installation, setup, and support

The ExerSights app can be accessed through the browser using the following URL https://exersights.web.app/. Users can opt to log in to ExerSights for access to additional configuration features like saving exercise angle preferences, pinning exercises, configuring/saving exercise programs, and saving/viewing exercise histories/summaries. However, to preserve full user anonymity, the app is fully functional if the user decides not to log in.

For first-time users learning how to navigate the pages, we provide a tutorial video at the bottom of the home page. If the user has points of confusion and needs support, we provide an FAQ page addressing common concerns and a contact form if they require further support. The contact form also allows users to submit feedback or requests for new exercises to be added to the catalog.

We also provide a Progressive Web App (PWA) option for mobile devices that also enables offline use. To download the PWA, go to the https://exersights.web.app/ URL and follow the instructions in the pop-up appearing at the bottom of the screen. Typically, it will instruct the user to press the ‘Share’ button, then the ‘Add to Home Screen’ button, where the user can customize what they would like to name the app, before adding it right onto their home screen.

The device requirements for running the app include having a functioning webcam as well as access to a browser capable of running JavaScript (most modern browsers like Chrome, Safari, Firefox, etc. are sufficient). An external webcam is optional but can be helpful if users do not want to use their device’s webcam.

3 Operation of the Project - Anish

3.1 Operating Mode 1: Normal Operation

In normal operation on laptops, users will open the app by navigating to https://exersights.web.app/. For mobile devices, if the user wishes to download and use the progressive web app, they must follow the browser instructions appearing at the bottom of the page during the initial entry.

Users will perform the following steps once on the home page:

1. Navigate to the Catalog page. Make sure you are logged in if you would like to use some of the account-specific features such as exercise settings, workout programs, and pinned exercises.
2. Search for and select a desired exercise by clicking on the exercise card.
3. Once on the exercise page, observe the feedback panel on the right. Click on the help (?) button to open a modal containing a tutorial and camera placement instructions for that specific exercise. Click the ‘close’ button or anywhere off the modal once finished viewing.
4. Click on the settings (gear) button to open a modal containing various exercise settings specifications including custom angle targets (such as target depth for squat) and voice selection for audio feedback.
5. If you would like to upload a video, click on the “Upload Video” button on the feedback panel.
6. Perform the exercise live by using your device’s webcam (or play your uploaded video) and hit the “Start Feedback” button. Ensure that your entire body is in frame and begin the exercise, adjusting your form based on the feedback.
    a. If you would like to switch to an external webcam which is different from your device’s default, use the dropdown next to the “Start Feedback” button.
    b. If you would like to use the timer, set a duration on the feedback panel and click the “Start n-Second Timer” (where n is the number of seconds you input in the duration field; 30 seconds by default) button to begin.
7. At the end of the exercise, a summary pop-up will be displayed. If you are logged in, you will have the option to save this summary which can be viewed later in the “My ExerSights” page at the top right.

Users also have the ability to create personalized workouts by performing the following:

1. Make sure you are logged in, and navigate to the Program page.
2. Click the + icon to create a new workout.
3. Click the gear icon to edit existing workouts. This will open a modal where you can select an exercise from the dropdown and add/position it within the program. Click “Save” and “Close” to save your changes to the database and exit out of the modal.

Users can navigate to the FAQ page to see answers to frequently-asked questions and a form to submit requests for new exercises, feedback, or additional questions.

3.2 Operating Mode 2: Abnormal Operations

Crowded Environment:

We do not advise using ExerSights in a crowded environment. While it is possible for the Mediapipe computer vision model to detect more than one person in the frame, we have limited control over the model and cannot guarantee that it will always detect the correct person performing the exercise. By default, ExerSights is programmed to assume only one person is in the frame. To avoid the model switching to another person, ensure that you are the only one in frame.

Lighting/Background:

To ensure the highest feedback accuracy, please perform exercises in a well-lit area where your body joints/limbs are discernible and visible to the camera. Also, make sure that there is a degree of contrast between you and the background. If you are wearing all blue clothes and your background is a blue wall, the model may have difficulty identifying your joint positions.

Camera Placement:

Each exercise has its own ideal camera placement, which ensures the highest accuracy of its feedback. For example, the squat exercise recommends users to position themselves so that the side of their body is facing the camera. Although it is possible to perform the exercise with the front of your body facing the camera, it will not be as accurate.

Visibility of Joints/Limbs:

In order to receive the most accurate feedback, please ensure that your entire body is visible to the camera. It is possible for the computer vision model to guess the position of non-visible joints, but this is often inaccurate and jittery. If your entire body is not in frame, the feedback panel will display the “Make sure all limbs are visible” error message as a warning.

Cheating the Model:

It is possible to cheat while doing many of the exercises, as our system takes into account a limited number of joint angles/positions to calculate feedback. For example, you can cheat the squat exercise by simply lifting your knee up to bend it beyond the target angle. We assume that all users are acting in good faith when using the app. Cheating the app is not advised as it disrupts your own workout efficacy.

3.3 Safety Issues

One of the top goals of the app is to make workouts safer by prompting users to correct bad/dangerous forms. We treat safety as a top priority, which is why we urge users to view/read the tutorial (help modal) before beginning any exercise. We also urge users to be cautious when changing the settings (target angle values) for any exercise. This being said, the computer vision model can sometimes be jittery or inaccurate. We provide a disclaimer on our home page: “Exercise feedback provided may not always be accurate. We are not liable for inaccurate feedback or any resulting injuries. Always consult a professional trainer before performing exercises. Use at your own risk.”

4 Technical Background - Jilin

ExerSights is built on top of three main technologies: React.js, MediaPipe Solutions, and Firebase.

React.js (referred to as React from here onwards) is one of the most popular frontend web development libraries. Aside from having a huge community and ecosystem, with tons of resources online for developers, React has two technical benefits that give it the edge over competing libraries/frameworks. First, React uses a virtual Document Object Model (DOM). The DOM is the programming interface for web documents that represent elements of a page as nodes and the page itself as a tree of nodes. Using the virtual DOM, React updates only the necessary parts of the real DOM, allowing for dynamic updates without having to reload the entire page, improving overall performance. Second, React has a component-based architecture, where components that have their own states can be easily reused across different parts of the web application, enhancing readability, maintainability, and scalability.

The entire frontend user interface of ExerSights is built on React. As mentioned previously, React uses a component-based architecture. Naturally, with the wide community behind React, many *component libraries* arise, further simplifying the development process by allowing us as developers not to have to re-implement common user interfaces. In our case, we opted for the Material UI (MUI) component library, a highly customizable, comprehensive React component library that implements Google's Material Design and ships production-ready components such as Button, Typography, and Card, all of which are ubiquitous throughout ExerSights. MUI was also designed as a mobile-first library, providing tools like Breakpoints and useMediaQuery to help create responsive user interfaces, which we certainly utilized to provide the best user experience.

At the core of ExerSights is human pose (description of a body's position at a moment in time with a set of skeletal landmark points) detection, achieved through Google's MediaPipe Solutions. Mediapipe Solutions is Google's suite of artificial intelligence (AI) and machine learning (ML) libraries that are quickly deployable for developers to utilize AI/ML in their applications. Mediapipe offers many different multimedia processing "tasks," optimized for low latency and real-time processing, with multiple flavors of models per task that trade off between performance and accuracy. For ExerSights, we utilize the pose landmark detection computer vision task and choose the least resource-intensive, lite model to minimize the compute resources needed client-side while providing the lowest latency real-time feedback to users. As for us developers, the pose landmark detection task gives us a straightforward application programming interface (API) to easily instantiate the model and begin processing camera feeds or uploaded videos in real-time. Upon the model's completion of processing one frame, the API returns a list of the detected human pose's landmarks, allowing us to perform the necessary angle calculations to determine the current state of the user.

It would be ideal if the Mediapipe AI/ML task could also tell us from which previous state to which new state the user transitioned from, but that is not the case. Instead, we use a unique finite state machine (FSM) for each exercise that dictates from which part of an exercise a user can transition to and vice-versa. This method enables us to accurately track when a user has completed a repetition of the specific exercise. Take the squat exercise, for example, which uses three main states: ‘descending,’ ‘squatting’, and ‘standing’. The only way the user can complete a repetition is by moving from the ‘descending’ state, past a certain exercise-specific threshold, to the ‘squatting’ state. After the ‘squatting’ state, the user must move, again, past a certain threshold, to transition into the ‘standing’ state. From the ‘standing’ state, once again, the user must move past a certain threshold to transition to the ‘descending’ state. The cycle continues. If the user has completed a squat and is in the ‘squatting’ state, and tries to cheat a subsequent repetition by not standing fully back up, they will **not** break the threshold to transition to the ‘standing’ state, and will not be able to move into the ‘descending’ state, as the transition into ‘descending’ can only be from ‘standing’. Note that to allow for any and everyone to use ExerSights, we have made these thresholds customizable, as every person’s anatomy is different. If a user finds discomfort with the default threshold, they can easily personalize the threshold as they see fit.

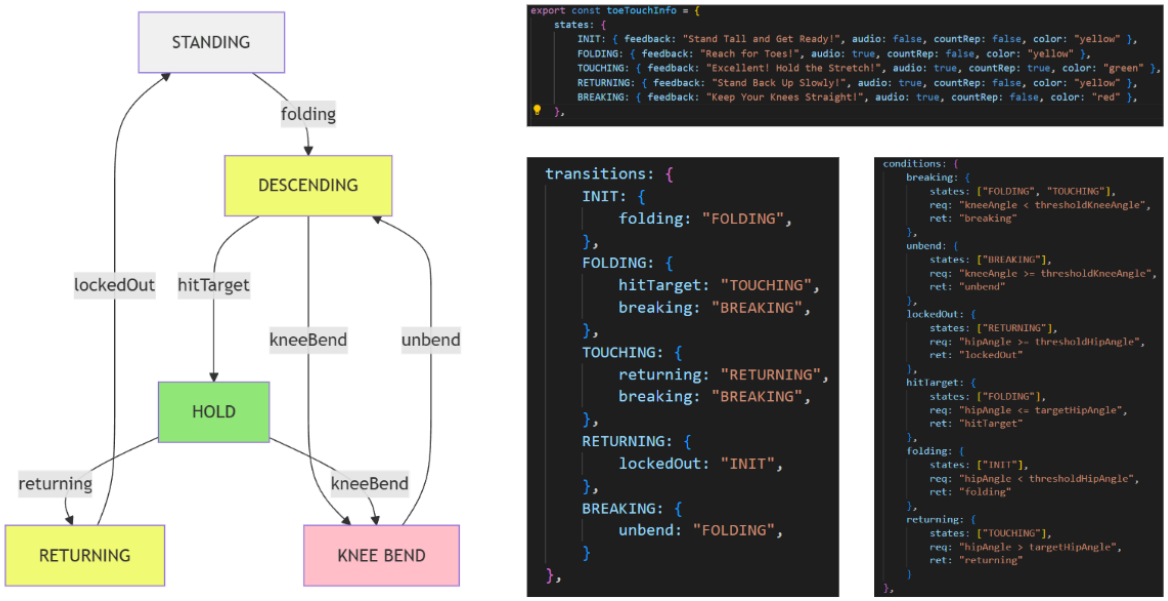

Figure 4.1 The finite state machine (FSM) diagram of the Standing Toe Touch exercise. The FSM is represented by the JSON object associated with the exercise. The JSON object specifies the states, transitions, and conditions of the FSM as it is passed to the generalized feedback calculation method.

Surprise! ExerSights does **not** have a **dedicated** backend infrastructure. And this is for good reason. To minimize latency for our real-time pose detection (beyond solely relying on the Mediapipe models being optimized for low latency), we deploy the Mediapipe task directly on the React front-end. Instead of incurring the overhead of setting up a dedicated back-end server ourselves, we utilize Firebase, Google’s “Backend-as-a-Service” (BaaS), which comes with a large suite of industry-grade

products that implement common features on web and mobile applications. Specifically for ExerSights, we use App Hosting, Authentication, Cloud Firestore, and Analytics. These Firebase products provide an extraordinarily developer-friendly API for us to quickly integrate these features into our app. App Hosting makes deploying and hosting our application incredibly streamlined, by synchronizing directly with the ‘main’ branch of our open-source GitHub repository. Furthermore, we set up a Continuous Integration/Continuous Deployment (CI/CD) pipeline to even deploy public links via App Hosting for our pull requests, allowing us to easily check if new features to be added conflict with existing features. With Authentication, we are able to take advantage of Google’s time-tested sign-on services in less than 10 lines of code. We use Cloud Firestore as our NoSQL database to intuitively store user data, allowing us to easily provide users with the extensive customization that is key to ExerSights. Finally, Analytics allows us to see user activity across our web app, enabling us to see how users are navigating our app and if the app is able to handle the user base.

5 Relevant Engineering Standards - Howell

As a software project, we need to ensure that our tech stack, codebase, and deployment processes meet relevant engineering standards. Our main GitHub branch was also the deployment branch for the public-facing application, hosted on Firebase. To maintain the integrity of this environment, we established a CI/CD pipeline aligning with principles outlined in IEEE 12207. This standard provides a framework for software life cycle processes, emphasizing structured development, maintenance, and operation. Every new feature starts as a GitHub issue and is developed on a separate branch. Once the feature was ready, the developer submitted a pull request (PR) and assigned reviewers.

To ensure code quality and security, aligning with IEEE 1012, each PR triggered the following automated GitHub Actions: a GitHub CodeQL scan for potential security vulnerabilities, and a Firebase preview deployment so reviewers could test the update without pulling or checking out the branch locally. This process ensured any deployed features were stable and secure. The IEEE 1012 standard helped us focus on independent testing and evaluation to ensure software products meet their intended use.

In regard to our actual code base and coding practices, we followed modular and reusable coding principles. Our exercise pages were refactored into a generalized structure, with the exercise customization coming via a JSON data structure. This structure allowed us to easily develop new exercises, improving scalability and maintainability. The web page components were built via React components, allowing us to customize and reuse previous components and pages for new features if needed. Our code is licensed under the open-source Apache-2.0 license.

When it came to team dynamics and communication, we followed the Agile Scrum framework. We held weekly meetings where each member updated one another on their progress for the week, got necessary feedback, and were delegated new tasks for the next week. Every meeting was documented as well for organization and accountability.

Ethical and transparent software development practices were central to our project, aligning with the considerations outlined in IEEE 7000. This standard provides a framework for addressing ethical concerns throughout system design. We heavily emphasized ethical and transparent software development practices throughout the project. Our app does not collect or store personal user data, except for optional customizations explicitly saved by the user into an account. Login functionality is entirely optional, and core features remain accessible without an account in case the users want total anonymity. To further demonstrate our transparency and community contribution, our codebase is totally open-source and available via GitHub. Our repository and code are licensed under an Apache-2.0 license.

We had to consider potential liabilities from injury when performing an exercise with our application. We included a disclaimer and an FAQ section to communicate to users the risks and recommended usage practices. Each exercise implementation was based on sources and references that can be found on our GitHub as well.

6 Cost Breakdown - James

Project Costs for Production of Beta Version (Next Unit after Prototype)				
Item	Quantity	Description	Unit Cost	Extended Cost (1 year)
1	5 GB/day	Firebase Download	$0.60/day	$219.00
2	100 GB	Firebase Storage	$0.026/GB	$2.60
3	1	IOS Developer Program	$99.00/year	$99.00
4	2	External Camera (for testing)	$19.99	$39.98
			Beta Version-Total Cost (for 1 year)	$360.58

The cost for developing our application is Firebase, which we use as our storage solution, and an external camera. When developing the app, we decided that 100GB of storage and 5GB of downloads per day would be sufficient for a user base of a thousand users for a small application. 100 GB of data storage costs us $2.60 at $0.026 per gigabyte. To download 5GB of data from the database to our application per day costs us $0.60 at $0.12 per gigabyte.

The calculations for the storage come from us assessing how much storage five users require. 339 MB for five users gives us 67.8 MB per person. For 1000 users, this means we will require 67.8 GB, and we account for extra by having 100GB of storage.

We also suggest that the user spend $19.99 to purchase a webcam, but this is not required. The suggested webcam can be found here. To develop a mobile application for the iOS platform, it will cost us an additional $99 to get an iOS Developer Program license.

7 Appendices - All

7.1 Appendix A - Specifications

Requirement	Target/Required Metric	Actual Metric
Feedback: Latency	< 100 ms	33.35 ms
Feedback: Accuracy (Joint Angle)	± 5°	± 2°
Feedback: Accuracy (Rep Count)	> 90 %	98%
Feedback: Visibility Distance	> 2 ft	5 ft
Availability	24/7	24/7
Security	Code scanning on GitHub	CodeQL Security Scanner in CI/CD
Personalized User Accounts	Accounts must have custom exercise settings	Custom exercise settings, workouts, and pinned exercises
Responsive UI	UI must be responsive to all device dimensions.	UI is responsive for laptop and mobile use.

7.2 Appendix B – Team Information

James Knee
James developed the (multi-camera) webcam detection logic, 2-player game mode, and styling. He will be working at MathWorks this summer as a software engineer intern.

Anish Sinha
Anish primarily developed the backend exercise feedback logic, audio feedback system, and database operations. He will be working as a software engineer at Red Hat after graduating this May.

Howell Xia
Howell developed the visual feedback display and workout programs features. He also initiated marketing efforts and obtained user feedback for the project. He will be working as a software engineer at MathWorks after graduating this May.

Jilin Zheng
Jilin primarily developed the frontend UI, progressive web app, user ticketing system, exercise summaries/histories, and database operations. He will be working at GE Vernova this summer as a manufacturing automation software engineer intern.