Software Requirements Specification for FitCoachAR: Real-Time Exercise Form Analysis System

Seyedmohamad Mirhosseininejad

April 20, 2026

Contents

Revision History

Date	Version	Notes
2026-01-30	1.0	First Draft for CAS 741
2026-02-18	1.1	Addressed Peer Review Feedback
2026-03-03	1.2	Addressed Dr. Smith's Feedback on SRS
2026-03-12	1.3	Addressed Yibing Mei's Peer Review feedback (traceability and formatting)
2026-04-15	1.4	Final-doc pass: unified "direct line of sight" terminology in GD_ElbowAngle
2026-04-15	1.5	Final Documentation release for CAS 741
2026-04-20	1.6	Addressed remaining items from Dr. Smith's SRS review (issue #4): expanded User Characteristics, added per-item rationale to System Constraints, rewrote Rationale per-topic.

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

symbol	**unit**	**SI**
$^\circ$	angle	degree (derived)
s	time	second
pixel	length	pixel (screen coordinates)

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with The choice of symbols was made to be consistent with the biomechanics and computer vision literature. The symbols are listed in alphabetical order.

symbol	**unit**	**description**
$\mathbf{P}$	-	A generic Point *(Note: removing bold p for scalar progress)*
p	-	Normalized progress (0 to 1)
θ	$^\circ$	Interior angle of a joint
θ_{high}	$^\circ$	Upper bound angle threshold
θ_{low}	$^\circ$	Lower bound angle threshold
θ_{thresh}	$^\circ$	Threshold angle for rep state transition
P	pixel	A point in 2D space (x, y)
$\vec{v}$	-	Vector connecting two keypoints
Is_Valid	-	Boolean flag for repetition validity

1.3 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
FitCoachAR	FitCoachAR
TM	Theoretical Model
RGB	Red Green Blue (Video format)
ROM	Range of Motion

2 Introduction

The following sections provide an overview of the software requirements for FitCoachAR.

2.1 Purpose of Document

The purpose of this document is to specify the functional and non-functional requirements for FitCoachAR. This document is intended to be used by the development team for implementation and verification, and by the stakeholders to ensure the system meets the biomechanical analysis needs.

2.2 Scope of Requirements

The scope of FitCoachAR is limited to the **2D analysis** of human movement using a single camera. The system currently focuses on the **Squat** and **Bicep Curl** exercises. It ignores 3D depth perception (z-axis) unless inferred, and assumes a stable camera position. The system calculates geometric angles based on projected keypoints and does not measure force, torque, or muscle activity.

2.3 Characteristics of Intended Reader

The intended reader is expected to have the educational background of an upper-level undergraduate engineering or computer science student, with knowledge of:

- Basic biomechanics (joint angles, flexion, extension).
- Software Engineering principles (SRS structure, typical of a senior level design course).
- Basic Linear Algebra (vectors, dot products).

2.4 Organization of Document

This document follows the Scientific Computing SRS template proposed by Smith and Lai Smith and Lai (2005). It starts with the System Description, followed by the specific physics/math models (TMs, DDs, IMs), and concludes with the Requirements.

3 General System Description

FitCoachAR is a real-time fitness coaching application that uses computer vision to analyze human movement during exercises. By leveraging a standard webcam, the system detects the user's body pose, calculates joint angles, and provides immediate feedback on exercise form. The goal is to help users perform exercises correctly, reducing injury risk and improving effectiveness, without requiring expensive equipment like motion capture suits.

3.1 System Context

FitCoachAR operates as a local application. The user interacts directly with the software via a webcam and a display screen.

- **User Responsibilities:** Ensure good lighting, wear distinguishable clothing, and position themselves fully within the camera frame.
- **FitCoachAR Responsibilities:** Detect the user, track keypoints, calculate angles, and provide feedback in real-time.

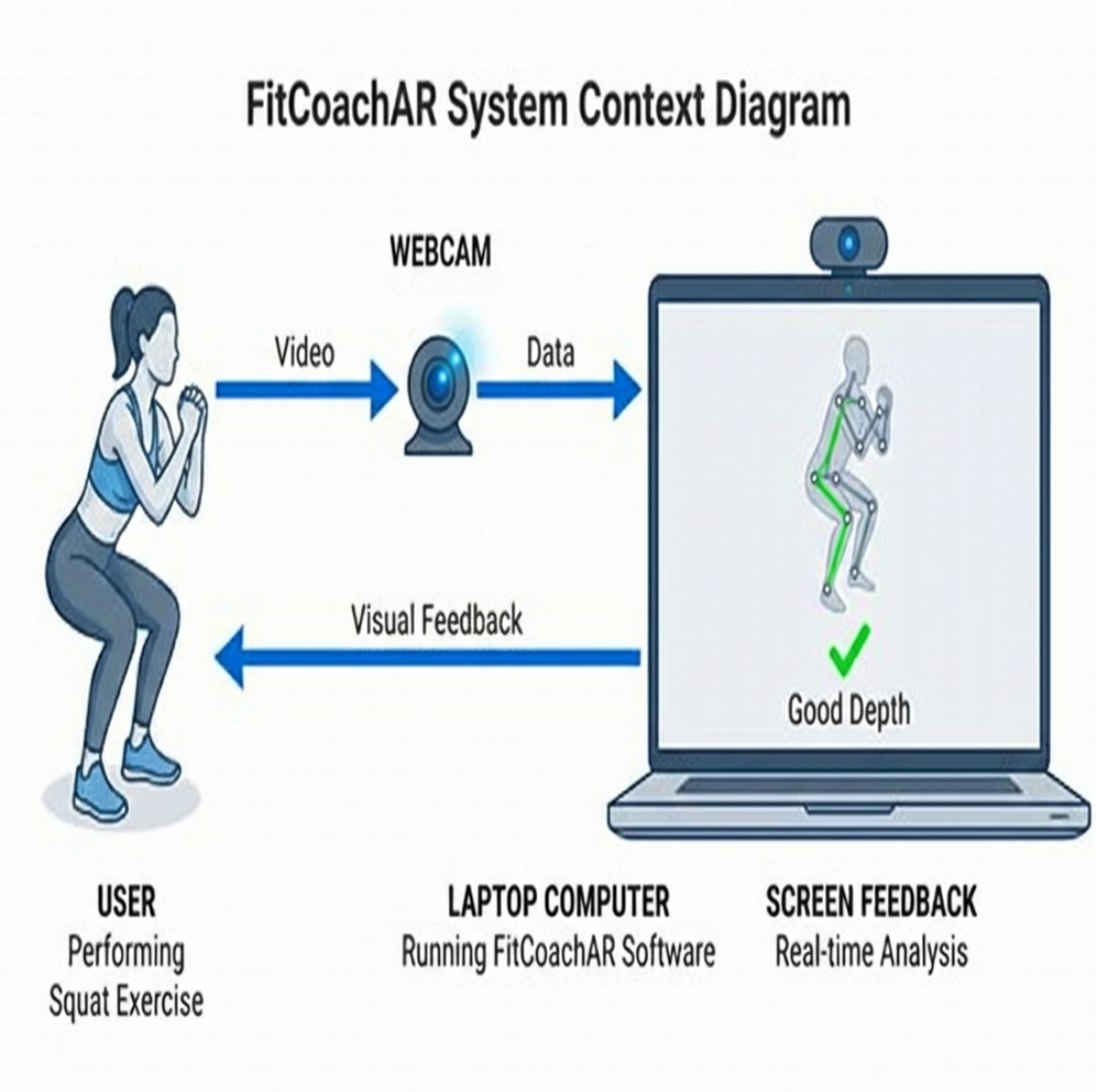

Figure 1: System Context: User → Webcam → FitCoachAR → Screen

3.2 User Characteristics

There are two groups of users for FitCoachAR.

End users are fitness enthusiasts or patients in physical therapy. They do not need any biomechanics, math, or programming knowledge. To use the software they just need to open a web browser, allow camera access, and read short text on the screen.

Readers of this SRS (reviewers, maintainers) are expected to have some background at roughly the first-year undergraduate level in engineering or computer science:

- Basic linear algebra: vectors, dot product, and norm.
- Basic biomechanics terms: joint, flexion, extension, range of motion. A first-year kinesiology course or general familiarity is enough; clinical training is not needed.
- Software engineering basics: what an SRS is and the difference between requirements and design, at the level of an upper-year undergraduate or first-year graduate software engineering course.

3.3 System Constraints

Only the decisions below are true requirements-level constraints. They are fixed by the problem and cannot be pushed to the design document without changing the requirements. A short rationale is given for each so it is clear why it is here and not later.

- The system is deployed as a **web application.** *Rationale:* the intended users should be able to run FitCoachAR on a laptop or phone they already own, without installing any software. This is a property of the problem, not a design choice. The specific web stack used to satisfy it (Django, WebSocket, a React front-end) is decided in the design document.
- The system uses an off-the-shelf **pose-estimation backend** (for example, MediaPipe or MoveNet) instead of a custom-trained model. *Rationale:* training a pose model from scratch is not realistic in a one-term project, and a single-subject dataset would not justify it anyway. This choice also puts a limit on the accuracy the system can reach (see NFR1), which is why it belongs with the requirements. The specific backend chosen is a design decision and is recorded in the Module Guide.

4 Specific System Description

4.1 Problem Description

Improper form during exercises like Squats can lead to injury and reduced effectiveness. FitCoachAR aims to solve this by providing automated, real-time coaching feedback without the need for expensive motion capture suits.

4.1.1 Physical System Description

The physical system includes:

PS1: The **User**, whose body joints (Hip, Knee, Ankle for legs; Shoulder, Elbow, Wrist for arms) are the targets of measurement.

PS2: The **Video Stream**, which captures the 2D projection of the User's movement.

4.1.2 Goal Statements

GS1: Analyze user video input to detect skeletal landmarks (Legs and Arms) in 2D space ($\mathbb{R}^2$).

GS2: Verify movement execution against established biomechanical thresholds (e.g., knee interior angle, elbow extension).

GS3: Count valid repetitions.

4.2 Solution Characteristics Specification

4.2.1 Assumptions

A1: The 2D projection of the user's side profile is sufficient to approximate the true biomechanical angles of the knee for a Squat.

A2: The camera remains stationary during the exercise.

A3: The relevant joints (Hip, Knee, Ankle for Squats; Shoulder, Elbow, Wrist for Bicep Curls) have a direct line of sight to the camera and are not obstructed by clothing or equipment.

A4: The environment has adequate lighting for the camera to distinguish the user from the background.

A5: The user's camera provides a minimum of 720p resolution for sufficient keypoint tracking accuracy.

4.2.2 Theoretical Models

This section focuses on the general equations and laws that FitCoachAR is based on.

RefName: **TM:VectorAngle**

Label: Calculation of Interior Angle between Two Vectors

Equation: $\theta = \arccos\left(\frac{\vec{v}_1 \cdot \vec{v}_2}{|\vec{v}_1||\vec{v}_2|}\right)$

Description: This model defines the interior angle θ between two vectors $\vec{v}_1$ and $\vec{v}_2$ in Euclidean space. The dot product $\cdot$ and the magnitude $|.|$ are standard vector operations.

Notes: None.

Source: Standard Linear Algebra Textbooks

Ref. By: GD1

Preconditions for TM:VectorAngle: Non-zero magnitude vectors.

Derivation for TM:VectorAngle: Not Applicable

4.2.3 Data Definitions

Number	DD1
Label	**Skeletal Keypoints (Legs)**
Symbol	P_H, P_K, P_A
SI Units	pixel (x, y coordinates)
Description	P_H: Coordinates of the Hip joint. P_K: Coordinates of the Knee joint. P_A: Coordinates of the Ankle joint. These are provided by the pose estimation backend.
Sources	Standard Biomechanics
Ref. By	GD1

Number	DD2
Label	**Skeletal Keypoints (Arms)**
Symbol	P_S, P_E, P_W
SI Units	pixel (x, y coordinates)
Description	P_S: Coordinates of the Shoulder joint. P_E: Coordinates of the Elbow joint. P_W: Coordinates of the Wrist joint. These are provided by the pose estimation backend.
Sources	Standard Biomechanics
Ref. By	GD2

4.2.4 General Definitions

This section collects the laws and equations that will be used in building the instance models.

Number	GD1
Label	**Knee Joint Angle Calculation**
SI Units	$^\circ$
Equation	$\theta_{knee} = \text{TM:VectorAngle}(\vec{v}_{femur}, \vec{v}_{tibia})$
Description	The knee angle is calculated using TM4.2.2. $\vec{v}_{femur} = P_{hip} - P_{knee}$. $\vec{v}_{tibia} = P_{ankle} - P_{knee}$. This assumes the 2D plane projection (A1).
Ref. By	IM1

Number	GD2
Label	**Elbow Joint Angle Calculation**
SI Units	$^\circ$
Equation	$\theta_{elbow} = \text{TM:VectorAngle}(\vec{v}_{humerus}, \vec{v}_{radius})$
Description	The elbow angle is calculated using TM4.2.2. $\vec{v}_{humerus} = P_{shoulder} - P_{elbow}$. $\vec{v}_{radius} = P_{wrist} - P_{elbow}$. This assumes a direct line of sight to the arm joints (A3).
Ref. By	IM2, IM3

Number	GD3
Label	**Normalized Repetition Progress**
SI Units	Dimensionless (0..1)
Equation	$p = \frac{\theta - \theta_{low}}{\theta_{high} - \theta_{low}}$
Description	Maps the current angle θ to a progress value $p \in [0, 1]$. θ_{low} and θ_{high} are calibrated ROM limits for the user. If $p \approx 0$, the user is at the start (eccentric start). If $p \approx 1$, the user is at the peak (concentric end).
Ref. By	IM3

4.2.5 Instance Models

This section details the instance models that use the general definitions to provide the necessary solution.

Number	IM1
Label	**Squat Depth Validity Criteria**
Input	θ_{knee} (°)
Output	$IsValid \in \mathbb{B}$
Description	$IsValid = (\theta_{knee} \geq \theta_{thresh\_squat})$. The squat is considered valid if the knee angle meets or exceeds the threshold. θ_{thresh_squat} is a default expected value (e.g., 90°).
Sources	Standard Squat Biomechanics
Ref. By	R4

Number	IM2
Label	**Bicep Curl Validity Criteria**
Input	θ_{elbow} (°)
Output	$IsValid \in \mathbb{B}$
Description	$IsValid = (\theta_{elbow} \geq \theta_{curl\_ext}$ AND $\theta_{elbow} \leq \theta_{curl\_flex})$. A repetition is valid if the arm starts at full extension and curls to full flexion. Default conservative values are used for extension and flexion.
Sources	Standard Bicep Curl Biomechanics
Ref. By	R4

Number	IM3
Label	**Hysteresis-based Finite State Machine for Rep Counting**
Input	p (from GD3), t (time)
Output	$RepCount$ ($\mathbb{N}$)
Description	A full repetition is counted ($RepCount \leftarrow RepCount + 1$) when the state machine completes the cycle: 1. **Start**: $p \leq 0.15$ (Bottom State) 2. **Ascend**: p increases > 0.15 (Up Phase) 3. **Peak**: $p \geq 0.85$ (Top State) 4. **Descend**: p decreases < 0.85 (Down Phase) 5. **Complete**: $p \leq 0.15$ (Return to Bottom) Constraints: The full cycle duration Δt must satisfy $t_{min} \leq \Delta t \leq t_{max}$.
Sources	Discrete Event Systems, Signal Processing
Ref. By	R5

5 Requirements

5.1 Functional Requirements

R1: The system shall accept real-time video input from a webcam, and the user-selected exercise type.

R2: The system shall track skeletal keypoints for legs (DD1) and arms (DD2) in each frame.

R3: The system shall calculate the knee angle θ_{knee} using GD1 and elbow angle θ_{elbow} using GD2.

R4: The system shall determine the validity of the exercise based on the selected exercise input: IM1 (for Squats) or IM2 (for Bicep Curls).

R5: The system shall display the skeletal overlay and feedback text on the screen.

5.2 Nonfunctional Requirements

NFR1: **Accuracy:** The calculated angles shall be within $\pm 5^\circ$ of visual ground truth for frames where the pose extraction confidence score exceeds the backend's reliability threshold (i.e., non-occluded). Verification procedures are detailed in the V&V Plan.

NFR2: **Portability:** The system shall be platform-independent, accessible on any device with a modern web browser and camera.

5.3 Rationale

This section explains why the scope, the assumptions, and the constraints were chosen.

Scope. FitCoachAR covers two exercises (squat and bicep curl) using a single camera and 2D analysis. Both exercises are common, and both have well-known biomechanical criteria that can be written as simple validity rules (see IM1 and IM2). Sticking to a single camera and 2D follows directly from the goal of running on hardware the user already owns; anything that needs a depth sensor, an IMU, or multiple cameras would shut out the intended users. Force, torque, and muscle activity are left out for the same reason: measuring them needs equipment the target user does not have.

Assumptions. Each assumption lets us keep the mathematical model simple, and each one has a failure mode that the system has to recognise at runtime.

- A1 (2D projection is enough for squat knee angle) makes it possible to use TM4.2.2 directly on image-plane coordinates. It fails when the camera is off-axis, which is why we pair it with A2 and the "side profile" user responsibility in the System Context.
- A2 (stationary camera) lets us treat the camera frame as a fixed coordinate system. Allowing a moving camera would need ego-motion estimation, which is a different problem.
- A3 (unobstructed joints) is what lets us use the pose-estimation library as a black box. Occlusion is the failure mode, and it is handled through the per-landmark visibility score. Test T9 in the VnV Plan verifies this rather than assuming the assumption always holds.
- A4 and A5 are conditions the user needs to provide. When either fails (dark scene, low resolution), the backend confidence score drops and the affected frames are skipped, so the failure is detected instead of silently corrupting the output.

Constraints. The two constraints in Section 3.3 are requirements-level, not design-level. Being a web application is what defines who can use the tool: anyone with a browser and a webcam. Relying on an off-the-shelf pose-estimation backend places an upper bound on the accuracy the system can claim (R1), because most of the error comes from the backend, not from the math that runs on top of it. The dynamic benchmark in VnV Report §4.1 confirms this *a posteriori*. Stating the constraint here makes that accuracy bound visible before the reader reaches the design document.

Trade-off. All of these choices trade specificity for reach. A system using 3D motion capture, clinical biomechanical models, or a multi-subject trained network would give better accuracy, but it would also rule out the users we want to serve. What is kept here is the minimum that still lets the tool be usable on an ordinary laptop while supporting a defensible accuracy claim.

6 Likely Changes

LC1: The thresholds for squat depth (θ_{thresh_squat}) and bicep curl range (θ_{curl_ext}, θ_{curl_flex}) are currently hardcoded, but may become dynamically customizable via user calibration routines in the future.

LC2: Additional exercises (e.g., Lunges, Shoulder Press) may be added, requiring new IMs.

7 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well. Table 2 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 3 shows the dependencies of instance models, requirements, and data constraints on each other. Table 1 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

The purpose of the traceability graphs is also to provide easy references on what has to be additionally modified if a certain component is changed. The arrows in the graphs represent dependencies. The component at the tail of an arrow is depended on by the component at the head of that arrow. Therefore, if a component is changed, the components that it points to should also be changed.

	A1	A2	A3
GD1	X	X	
GD2	X	X	X
GD3	X	X	
IM1	X	X	
IM2	X	X	
IM3	X	X	

Table 1: Traceability Matrix Showing the Connections Between Assumptions and Other Items

	TM:VectorAngle	DD1	DD2	GD1	GD2	GD3
GD1	X	X				
GD2	X		X			
GD3						
IM1				X		
IM2					X	
IM3						X

Table 2: Traceability Matrix Showing the Connections Between Items of Different Sections

	A2	DD1	DD2	GD1	GD2	GD3	IM1	IM2
R1	X							
R2	X	X	X					
R3				X	X	X		
R4							X	X
R5							X	X

Table 3: Traceability Matrix Showing the Connections Between Requirements and Instance Models

8 Development Plan

Metric-based verification will be prioritized in the first iteration. A detailed schedule and testing strategy can be found in the Verification and Validation (V&V) Plan.

9 Values of Auxiliary Constants

The following symbolic constants represent the default biomechanical thresholds assumed by the system for ideal form.

- $\theta_{thresh\_squat} = 90^\circ$ (Minimum depth angle for a valid squat)
- $\theta_{curl\_ext} = 160^\circ$ (Minimum extension angle for a valid bicep curl startup)
- $\theta_{curl\_flex} = 30^\circ$ (Maximum flexion angle at the peak of a valid bicep curl)

References

W. Spencer Smith and Lei Lai. A new requirements template for scientific computing. In J. Ralyté, P. Àgerfalk, and N. Kraiem, editors, *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, pages 107–121, Paris, France, 2005. In conjunction with 13th IEEE International Requirements Engineering Conference.