Module Interface Specification for FitCoachAR

Seyedmohamad Mirhosseininejad

April 21, 2026

1 Revision History

Date	Version	Notes
2026-03-23	1.0	Initial MG/MIS presentation draft
2026-04-02	1.1	Full MIS formalization aligned with 8-module implementation
2026-04-07	1.2	Reviewer's edit (Yibing Mei and Dr. Smith)
2026-04-15	1.3	Final-doc consistency pass: corrected M8 "Uses" from M4 to None
2026-04-15	1.4	Final Documentation release for CAS 741
2026-04-20	1.5	Added Module Kind classification (AO/ADT/Library) to each module.
2026-04-20	1.6	M2 reclassified as Library (no state); added initialisation assumption to M4.

2 Symbols, Abbreviations and Acronyms

Symbol	Description
MIS	Module Interface Specification
MG	Module Guide
SRS	Software Requirements Specification
ABC	Abstract Base Class (Python interface enforcement)
M1–M8	Module identifiers as defined in the MG
$\mathbb{R}$	Real number
$\mathbb{Z}$	Integer
$\mathbb{N}$	Natural number
Point2D	A tuple $(x, y) \in \mathbb{R}^2$ representing image coordinates
Point3D	A tuple $(x, y, z) \in \mathbb{R}^3$ representing 3D coordinates
JointID	An integer index identifying a MediaPipe pose landmark
LandmarkList	A sequence L of 33 records, where each $L_i = \{x : \mathbb{R}, y : \mathbb{R}, z : \mathbb{R}, v : \mathbb{R}\}$
ExerciseEnum	{SQUAT, BICEP_CURL, UNKNOWN}

Contents

3 Introduction

The following document details the Module Interface Specifications for FitCoachAR, an augmented reality application that provides real-time biomechanical feedback for fitness exercise using a standard webcam.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/mmdmirh/cas741.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). The symbol := is used for assignment and conditional rules follow the form $(c_1 \Rightarrow r_1 \mid c_2 \Rightarrow r_2)$.

The following table summarizes the primitive data types used by FitCoachAR.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in $(-\infty, \infty)$
natural number	$\mathbb{N}$	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$
boolean	$\mathbb{B}$	`True` or `False`

Derived types used include: sequences (ordered lists of same-type elements), tuples (fixed-length heterogeneous collections), and functions (typed input/output mappings). Interfaces are enforced in Python via Abstract Base Classes (ABCs) from the `abc` module.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2
Hardware-Hiding	M1: Video Input Module
Behaviour-Hiding	M2: Display Output Module M3: Video Formatting Module M4: Pose Tracking Module M5: Kinematic Engine Module M6: Exercise State Module M7: UI Rendering Module M8: Signal Smoothing Module
Software Decision	–

Table 1: Module Hierarchy

6 MIS of M1: Video Input Module

6.1 Module

Browser-native Video Capture API (e.g., `getUserMedia`)

6.2 Module Kind

Library (hardware-hiding wrapper; no client-visible state, delegates to the browser's `getUserMedia` service).

6.3 Uses

None.

6.4 Syntax

6.4.1 Exported Access Programs

Name	In	Out	Exceptions
`get_frame`	–	frame: bytes	`CameraError`

6.5 Semantics

6.5.1 State Variables

None.

6.5.2 Environment Variables

Input from the hardware webcam sensor.

6.5.3 Access Routine Semantics

`get_frame()`:

- output: The current image frame captured from the webcam, as a byte stream in JPEG/PNG format.
- exception: `CameraError` if the device is unavailable or access is denied.

7 MIS of M2: Display Output Module

7.1 Module

Browser Canvas Rendering Module (`frontend/modules/m2_display_output.js`)

7.2 Module Kind

Library. The module has only behaviour (rendering side effects on the browser canvas) and no internal state; the canvas itself lives in the DOM and is treated as an environment variable.

7.3 Uses

None.

7.4 Syntax

7.4.1 Exported Access Programs

Name	In	Out	Exceptions
`draw_overlay`	landmarks: LandmarkList	–	–
`display`	frame: bytes	–	–

7.5 Semantics

7.5.1 State Variables

None.

7.5.2 Environment Variables

The browser DOM, specifically the 2D rendering context of the AR canvas element.

7.5.3 Access Routine Semantics

`draw_overlay(landmarks)`:

- output: Renders a connected skeleton on the AR canvas using the provided landmark coordinates.

`display(frame)`:

- output: Renders the raw video frame as the background of the AR canvas.

8 MIS of M3: Video Formatting Module

8.1 Module

`m3_video_formatting` (`backend/modules/m3_video_formatting.py`)

8.2 Module Kind

Library. Stateless encode/decode between raw image bytes and the Base64 transport representation.

8.3 Uses

M1 (Video Input)

8.4 Syntax

8.4.1 Exported Access Programs

Name	In	Out	Exceptions
`encode_frame`	frame: bytes	encoded: str	`InvalidImageEx`
`decode_frame`	encoded: str	frame: bytes	`Base64Ex`

8.5 Semantics

8.5.1 State Variables

None.

8.5.2 Environment Variables

None.

8.5.3 Assumptions

Input bytes represent a valid compressed image (JPEG or PNG).

8.5.4 Access Routine Semantics

`encode_frame(frame)`:

- output: Base64-encoded UTF-8 string representation of the raw image bytes.
- exception: $exc :=$ (frame is empty $\Rightarrow$ `InvalidImageEx` | True $\Rightarrow$ None)

`decode_frame(encoded)`:

- output: Raw image bytes decoded from the Base64 string.
- exception: $exc :=$ (encoded is not valid Base64 $\Rightarrow$ `Base64Ex` | True $\Rightarrow$ None)

9 MIS of M4: Pose Tracking Module

9.1 Module

`m4_pose_tracking` (`backend/modules/m4_pose_tracking.py`)

9.2 Module Kind

Abstract Object. A single loaded pose-estimation backend instance is held as internal state; the module is imported and used directly rather than instantiated by clients.

9.3 Uses

M3 (Video Formatting), M8 (Signal Smoothing)

9.4 Syntax

9.4.1 Exported Constants

`NUM_LANDMARKS` := 33 (number of MediaPipe Pose landmarks)

9.4.2 Exported Access Programs

Name	In	Out	Exceptions
`detect`	frame: bytes	LandmarkList	`DetectionError`
`is_visible`	landmarks: LandmarkList, id: JointID	$\mathbb{B}$	-

9.5 Semantics

9.5.1 State Variables

- `model`: Loaded ML inference model instance.

9.5.2 Environment Variables

None.

9.5.3 Assumptions

- The input frame contains at most one person.
- Lighting conditions meet minimum hardware requirements.

- **Initialisation.** The backend is loaded the first time the module is imported (Python module-level import performs the `load_model` step). Programmers using this module must therefore import `m4_pose_tracking` before calling `detect` or `is_visible`.

9.5.4 Access Routine Semantics

`detect(frame)`:

- input: raw bytes of a JPEG/PNG image.
- output: A sequence L of 33 records (LandmarkList).
- exception: $exc :=$ (no person detected $\Rightarrow$ `DetectionError` | True $\Rightarrow$ None)

`is_visible(landmarks, id)`:

- output: `True` iff `landmarks[id].v` > 0.5.

10 MIS of M5: Kinematic Engine Module

10.1 Module

`m5_kinematic_engine` (`backend/modules/m5_kinematic_engine.py`)

10.2 Module Kind

Library. Stateless computation: joint-angle and distance calculations from 3D point inputs, with no state kept between calls.

10.3 Uses

M4 (Pose Tracking)

10.4 Syntax

10.4.1 Exported Constants

None.

10.4.2 Exported Access Programs

Name	In	Out	Exceptions
`compute_angle`	a, b, c: Point3D	$\mathbb{R}$	`CollinearEx`
`extract_metrics`	l: LandmarkList, e: ExerciseEnum	record	`TypeEx`

10.5 Semantics

10.5.1 State Variables

None. (Stateless computation module.)

10.5.2 Environment Variables

None.

10.5.3 Assumptions

- All three points a, b, c are non-collinear (i.e., $\|a - b\| > 0$ and $\|c - b\| > 0$).
- `exercise` string is one of the recognized exercise types (e.g., `"squat"`, `"curl"`).

10.5.4 Access Routine Semantics

`compute_angle(a, b, c)`:

- output: The angle θ at joint b formed by vectors $\overrightarrow{ba}$ and $\overrightarrow{bc}$, in degrees:

$$\theta = \arccos\left(\frac{\overrightarrow{ba} \cdot \overrightarrow{bc}}{\|\overrightarrow{ba}\| \cdot \|\overrightarrow{bc}\|}\right) \times \frac{180}{\pi}$$

  where $\theta \in [0, 180]$.

- exception: $exc :=$ ($\|\overrightarrow{ba}\| = 0 \vee \|\overrightarrow{bc}\| = 0 \Rightarrow$ `CollinearEx` | True $\Rightarrow$ None)

`extract_metrics(l, e)`:

- output: A record $r = \{primary : \mathbb{R}, form\_vals : \text{seq of } \mathbb{R}\}$ where:
  - $r.primary =$ (e = SQUAT $\Rightarrow$ `compute_angle`(_get_coords(l, 24), _get_coords(l, 26), _get_coords(l True $\Rightarrow$ 0)
  - $r.form\_vals$ is a sequence of angles computed using `compute_angle` for secondary joints.
- exception: $exc :=$ (e = UNKNOWN $\Rightarrow$ `TypeEx` | True $\Rightarrow$ None)

10.5.5 Local Functions

`_get_coords(landmarks, id: JointID)` $\rightarrow$ `Point3D`:

- Returns (x, y, z) for landmark at index `id`.

`_compute_distance(p1, p2: Point3D)` $\rightarrow \mathbb{R}$:

- Returns Euclidean distance $\|p1 - p2\|$.

11 MIS of M6: Exercise State Module

11.1 Module

`m6_exercise_state` (`backend/modules/m6_exercise_state.py`)

11.2 Module Kind

Abstract Data Type. Each exercise session creates its own `ExerciseStateMachine` instance, each with its own `current_state`, `rep_count`, and per-rep angle bounds.

11.3 Uses

M5 (Kinematic Engine)

11.4 Syntax

11.4.1 Exported Constants

- `STATES` := {`BOTTOM`, `UP_PHASE`, `TOP`, `DOWN_PHASE`}

11.4.2 Exported Access Programs

Name	In	Out	Exceptions
`update`	angle: $\mathbb{R}$, dt: $\mathbb{R}$	state: str	-
`get_rep_count`	-	$\mathbb{N}$	-
`reset`	-	-	-

11.5 Semantics

11.5.1 State Variables

- `current_state`: str $\in$ `STATES` (initially `BOTTOM`)
- `rep_count`: $\mathbb{N}$ (initially 0)
- `angle_min`: $\mathbb{R}$ (initially ∞)
- `angle_max`: $\mathbb{R}$ (initially $-\infty$)

11.5.2 Environment Variables

None.

11.5.3 Assumptions

- Angles are produced by M5 at a rate sufficient for state detection ($\geq$ 15 fps).
- Calibration parameters (ROM thresholds) are set before `update` is called.

11.5.4 Access Routine Semantics

`update(angle, dt)`:

- transition: Advances the state machine according to:

$$\texttt{BOTTOM} \xrightarrow{\text{angle} \geq \theta_{top\_min}} \texttt{UP\_PHASE}$$

$$\texttt{UP\_PHASE} \xrightarrow{\text{angle} \geq \theta_{top\_max}} \texttt{TOP}$$

$$\texttt{TOP} \xrightarrow{\text{angle} \leq \theta_{bottom\_max}} \texttt{DOWN\_PHASE}$$

$$\texttt{DOWN\_PHASE} \xrightarrow{\text{angle} \leq \theta_{bottom\_min}} \texttt{BOTTOM},\ \texttt{rep\_count} := \texttt{rep\_count} + 1$$

- output: The new `current_state` string after the transition.

`get_rep_count()`:

- output: Current value of `rep_count`.

`reset()`:

- transition: `rep_count` := 0, `current_state` := `BOTTOM`.

12 MIS of M7: UI Rendering Module

12.1 Module

FitCoachAR Web Interface Renderer (`frontend/modules/m7_ui_rendering.js`)

12.2 Module Kind

Library. Stateless JSON-to-DOM update; reads the current session state and emits UI updates without holding state of its own.

12.3 Uses

M6 (Exercise State)

12.4 Syntax

12.4.1 Exported Access Programs

Name	In	Out	Exceptions
`update_ui`	metrics: record	–	–
`get_json`	–	str	–

12.5 Semantics

12.5.1 State Variables

None.

12.5.2 Environment Variables

Browser Application State / DOM elements.

12.5.3 Access Routine Semantics

`update_ui(metrics)`:

- output: Updates the rep count and stability badges in the sidebar using values from `metrics`.

`get_json()`:

- output: A JSON-encoded string representation of the current session metrics for transmission.

13 MIS of M8: Signal Smoothing Module

13.1 Module

`m8_signal_smoothing` (`backend/modules/m8_signal_smoothing.py`)

13.2 Module Kind

Abstract Data Type. Each smoothed channel (one per landmark coordinate, so 99 instances for a full 33-landmark skeleton) holds its own scalar Kalman state `x_est`, `P`, `Q`, `R`. The scalar types in §10.4.1 are therefore per-instance state, not global state.

13.3 Uses

None

13.4 Syntax

13.4.1 Exported Access Programs

Name	In	Out	Exceptions
`smooth`	measurement: $\mathbb{R}$	estimate: $\mathbb{R}$	-
`smooth_landmarks`	landmarks: LandmarkList	LandmarkList ($R^{33\times3}$)	-

13.5 Semantics

13.5.1 State Variables

- `x_est`: $\mathbb{R}$ (initially 0)
- `P`: $\mathbb{R}$ (initially 1)
- `Q`, `R`: $\mathbb{R}$ (noise covariances)

13.5.2 Environment Variables

None.

13.5.3 Assumptions

- Measurements are scalar real values from M4 at a fixed frame rate.

13.5.4 Access Routine Semantics

`smooth(measurement)`:

- transition: One-step Kalman filter update.
- output: Updated `x_est` (the filtered scalar estimate).

`smooth_landmarks(landmarks)`:

- output: A new LandmarkList where each (x, y, z) coordinate $c \in R^{33\times3}$ has been individually filtered.

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.