Module Interface Specification for SFWRENG 4G06 Capstone Design Project

Team #18, InfiniView-AI
Anhao Jiao
Kehao Huang
Qianlin Chen
Qi Shu
Xunzhou Ye

March 27, 2024

1 Revision History

Date	Notes
Jan 12	Add MIS for UI components
Jan 13	Add MIS for Media Control components, RTC Control components
Jan 14	Add MIS for Backend components
Jan 17	Revise before submission
Mar 27	Rev1

2 Symbols, Abbreviations and Acronyms

Symbol	Description
MG	Module Guide
M	Module
MIS	Module Interface Specification
HTTP	Hypertext Transfer Protocol
OS	Operating System
STUN	Session Traversal Utilities for NAT - a type of server needed for setting up peer-to-peer connections
RTC	Real-Time Communication
SFU	Selective Forwarding Unit - A software unit that can selectively forward video streams
API	Application Programming Interface
SDP	Session Description Protocol
WebRTC	Web Real-Time Communication
CM	Center of Mass Annotation Module
HPE	Human Pose Estimation Annotation Module

Contents

3 Introduction

The following document details the Module Interface Specifications for the MotionMingleapplication. Complementary documents include the Module Guide.

The full documentation and implementation can be found at MotionMingle.git.

4 Notation

The following tables summarize the primitive data types, derived data types, and other derived data types from aiortc, av, aiohttp, React and Web APIs libraries that are used by MotionMingle.

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). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

4.1 Primitive Data Types

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

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

The specification of MotionMingleuses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, MotionMingleuses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

4.2 Data Types from Libraries

The following table summarizes the data types provided by external libraries and used by MotionMingle.

Data Type	Notation	Description
VideoStreamTrack	VideoStreamTrack	A dummy video track that reads green frames.
MediaRelay	MediaRelay	A media source that relays one or more tracks to multiple consumers.
RTCPeerConnection	RTCPeerConnection	An interface represents a WebRTC connection between the local computer and a remote peer.
MediaStreamTrack	MediaStreamTrack	A single media track within a media stream.
RTCSessionDescription	RTCSessionDescription	An interface describes the potential connection and how it's configured. Each RTCSessionDescription consists of a description type indicating which part of the offer or answer negotiation process it describes and of the SDP descriptor of the session
JSON	JSON	JavaScript Object Notation, it is a text-based open standard data interchange setup and only provides a data encoding specification.
RTCTrackEvent	RTCTrackEvent	An event triggered by adding a MediaStreamTrack
MediaStream	MediaStream	A stream of data that usually carries media data
HTMLVideoElement	HTMLVideoElement	A react type representation of the video element in Hypertext Markup Language
React.component	React.component	A independent and reusable bits of react code that outputs HTML elements

Table 1: Data types from libraries

5 Module Decomposition

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

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	User Authentication Module Instructor View Module Practitioner View Module Annotation Configuration Module RTC Control Module
Software Decision Module	STUN Server Module App Module Video Transform Module Human Pose Estimation Annotation Module Center of Mass Annotation Module SFU Server Module

Table 2: Module Hierarchy

6 MIS of RTC Control Module

6.1 Module

RTCControl

6.2 Uses

Web APIs
STUN Server Module

6.3 Syntax

6.3.1 Exported Constants

N/A

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
createPeerConnection	JSON	RTCPeerConnection	-
closeRemoteConnection	RTCPeerConnection	-	-
negotiate	RTCPeerConnection	-	-

6.4 Semantics

6.4.1 State Variables

N/A

6.4.2 Environment Variables

STUN_SERVER_ADDRESS: string — represents the address of the STUN server
SFU_BROADCAST_API: string — represents the API endpoint for SFU broadcast API
SFU_CONSUME_API: string — represents the API endpoint for SFU consume API

6.4.3 Assumptions

SFU server and STUN servers are running in normal conditions.

6.4.4 Access Routine Semantics

createPeerConnectionWith(config: JSON):

- transition: N/A

- output: pc := RTCPeerConnection — initializes a new RTCPeerConnection based on the given configuration.
- exception: ~~N/A~~ ConfigurationError — thrown if the configuration is invalid or missing required information.

closeRemoteConneciton(pc: RTCPeerConnection):

- transition: pc.signalingState := closed — closes peer connection and send a signal to the connected peer connection.
- output: N/A
- exception: ~~N/A~~ InvalidStateError — thrown if the connection is already closed or not in a valid state to be closed.

negotiate(pc: RTCPeerConnection):

- transition:

  pc.localDescription := RTCSessionDescriptionInit

  pc.remoteDescription := RTCSessionDescriptionInit

  sets the local description of the peer connection to its generated SDP, and set the remote description of the peer connection to its received SDP from SFU_BROADCAST_API.
- output: N/A
- exception: ~~N/A~~ NegotiationError — thrown if the negotiation fails due to invalid SDP or connection state.

getRemoteStream(pc: RTCPeerConnection):

- transition: pc.event := getRemoteEvent(pc).streams
- output: N/A
- exception: ~~N/A~~ StreamNotFoundError — thrown if the remote stream cannot be found or is not accessible.

6.4.5 Local Functions

getRemoteEvent(pc: RTCPeerConnection):

- transition: N/A
- output: pc.event := RTCTrackEvent
- exception: ~~N/A~~ EventNotFoundError — thrown if the event related to the remote track is not found or is not triggered.

7 MIS of Media Control Module

7.1 Module

MediaContorl

7.2 Uses

Web APIs

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
setMicEnabled	Boolean	-	-
setCameraEnabled	Boolean	-	-
getStream	-	MediaStream	-

7.4 Semantics

7.4.1 State Variables

isMicEnabled: Boolean
isCameraEnabled: Boolean

7.4.2 Environment Variables

Microphone
Camera

7.4.3 Assumptions

User's devices have a functioning screen, camera and microphone.

7.4.4 Access Routine Semantics

setMicEnabled(isEnabled: Boolean):

- transition: isMicEnabled := isEnabled
- output: N/A
- exception: N/A

8 MIS of Media Control Module

8.1 Module

MediaContorl

8.2 Uses

Web APIs

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
setMicEnabled	Boolean	-	-
setCameraEnabled	Boolean	-	-
getStream	-	MediaStream	-

8.4 Semantics

8.4.1 State Variables

isMicEnabled: Boolean
isCameraEnabled: Boolean

8.4.2 Environment Variables

Microphone
Camera

8.4.3 Assumptions

User's devices have a functioning screen, camera and microphone.

8.4.4 Access Routine Semantics

setMicEnabled(isEnabled: Boolean):

- transition: isMicEnabled := isEnabled
- output: N/A

- exception: ~~N/A~~ DeviceAccessError — thrown if the microphone cannot be accessed or permissions are not granted.

setCameraEnabled(isEnabled: Boolean):

- transition: isCameraEnabled := isEnabled
- output: N/A
- exception: ~~N/A~~ DeviceAccessError — thrown if the microphone cannot be accessed or permissions are not granted.

getStream():

- transition: N/A
- output: returns the user media stream based on the state value isCameraEnabled and isMicEnabled
- exception: ~~N/A~~ DeviceAccessError — thrown if the microphone cannot be accessed or permissions are not granted.

8.4.5 Local Functions

N/A

9 MIS of Instructor View Module

9.1 Module

Instructor

9.2 Uses

Media Control Module
RTC Control Module
Annotation Configuration Module
React
Web APIs

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
Instructor	-	React.component	-

9.4 Semantics

9.4.1 State Variables

remoteVideoRef: HTMLVideoElement
selfVideoRef: HTMLVideoElement
peerConnection: RTCPeerConnection

9.4.2 Environment Variables

Screen

9.4.3 Assumptions

User's devices have a functioning screen, camera and microphone.

9.4.4 Access Routine Semantics

Instructor():

- transition: N/A
- output: renders a react component of the instructor view page
- exception: ~~N/A~~ RenderError — thrown if the component fails to render.

9.4.5 Local Functions

setPeerConnection(pc: RTCPeerConnection):

- transition: peerConnection := pc
- output: N/A
- exception: ~~N/A~~ ConnectionError — thrown if the peer connection cannot be established.

getSelfVideo():

- transition:

  selfVideoRef.current.video.srcObject:= MediaControl.getStream()

  render video stream from the local camera to screen

- output: N/A
- exception: ~~N/A~~ VideoStreamError — thrown if the local video stream cannot be accessed or is not available.

startRemoteSharing():

- transition: peerConnection.addTrack := MediaControl.getStream()
- output: N/A
- exception: ~~N/A~~ ShareStartError — thrown if the stream cannot be added to the peer connection or sharing cannot be initiated.

stopRemoteSharing():

- transition:

  remoteVideoRef.current.video.srcObject = null

  peerConnection.close:= true

  stops the remote video on the user's screen and close the RTCPeerConnection
- output: N/A
- exception: ~~N/A~~ ShareStopError — thrown if stopping the remote sharing fails or the connection cannot be closed.

getRemoteVideo():

- transition: get remote video coming from the SFU server and render it on the user's screen.
- output: N/A
- exception: ~~N/A~~ RemoteVideoError — thrown if the remote video cannot be retrieved or displayed.

10 MIS of Practitioner View Module

10.1 Module

Practitioner

10.2 Uses

Media Control Module
RTC Control Module
Annotation Configuration Module
React
Web APIs

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
Practitioner	-	React.component	-

10.4 Semantics

10.4.1 State Variables

remoteVideoRef: HTMLVideoElement
peerConnection: RTCPeerConnection

10.4.2 Environment Variables

Screen

10.4.3 Assumptions

User's devices have a functioning screen.

10.4.4 Access Routine Semantics

N/A

10.4.5 Local Functions

setPeerConnection(pc: RTCPeerConnection):

- transition: peerConnection := pc
- output: N/A

- exception: ~~N/A~~ ConnectionSetupFailure — thrown if the peer connection cannot be set due to an invalid or null ‘pc‘ argument, or if the connection setup fails.

getRemoteVideo():

- transition: get remote video coming from the SFU server and render it on the user’s screen.
- output: N/A
- exception: ~~N/A~~ VideoRetrievalError — thrown if the remote video cannot be retrieved from the SFU server, or if there is an error in rendering the video on the screen.

11 MIS of Annotation Configuration Module

11.1 Module

AnnotationConfig

11.2 Uses

RTC Control Module
React

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
setIsSkeletonEnabled	Boolean	-	-
setIsCOMEnabled	Boolean	-	-
getIsSkeletonEnable	-	Boolean	-
getIsCOMEnable	-	Boolean	-

11.4 Semantics

11.4.1 State Variables

isSkeletonEnabled: Boolean
isCOMEnabled: Boolean

11.4.2 Environment Variables

N/A

11.4.3 Assumptions

N/A

11.4.4 Access Routine Semantics

setIsSkeletonEnabled(isEnabled: Boolean):

- transition: isSkeletonEnabled := isEnabled
- output: N/A
- exception: ~~N/A~~ FeatureToggleError — thrown if there is an error in toggling the skeleton visualization state.

setIsCOMEnabled(isEnabled: Boolean):

- transition: isCOMEnabled := isEnabled
- output: N/A
- exception: ~~N/A~~ FeatureToggleError — thrown if there is an error in toggling the COM visualization state.

getIsSkeletonEnabled():

- transition: N/A
- output: isSkeletonEnabled
- exception: ~~N/A~~ StateRetrievalError — thrown if the current state of the skeleton feature cannot be retrieved.

getIsCOMEnabled():

- transition: N/A
- output: isCOMEnabled
- exception: ~~N/A~~ StateRetrievalError — thrown if the current state of the COM feature cannot be retrieved.

11.4.5 Local Functions

N/A

12 MIS of App Module

12.1 Module

App

12.2 Uses

RTC Control Module
Media Control Module
Instructor View Module
Practitioner View Module
Annotation Configuration Module
User Authentication Module

12.3 Syntax

12.3.1 Exported Constants

None

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
App	-	React.component	-

12.4 Semantics

12.4.1 State Variables

N/A

12.4.2 Environment Variables

N/A

12.4.3 Assumptions

N/A

12.4.4 Access Routine Semantics

App():

- transition: App:= React.component()
  starts React App and render it on the user's device
- output: N/A
- exception: N/A

12.4.5 Local Functions

N/A

13 MIS of User Authentication Module

13.1 Module

Auth

13.2 Uses

Instructor View Module
Practitioner View Module

13.3 Syntax

13.3.1 Exported Constants

N/A

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
Auth	-	React.component	-

13.4 Semantics

13.4.1 State Variables

isUserInstructor: Boolean

13.4.2 Environment Variables

N/A

13.4.3 Assumptions

N/A

13.4.4 Access Routine Semantics

Auth():

- transition: Render the authentication page on the user's device, if the user clicks on the Instructor button, then jumps to the instructor view page, if the user clicks on the practitioner button, jumps to the practitioner view page.
- output: N/A
- exception: N/A

13.4.5 Local Functions

isUserInstructor $\rightarrow$ Instructor view else Practitioner view
setIsUserInstructor(isEnabled: Boolean):

- transition: isUserInstructor := isEnabled
- output: N/A
- exception: N/A

14 MIS of Video Transform Module

14.1 Module

VideoTransformTrack

14.2 Uses

HPE, CM

14.3 Syntax

14.3.1 Exported Constants

kind = "video"

14.3.2 Exported Access Programs

Routine name	In	Out	Exceptions
__init__	track, transform	-	-
recv	-	VideoFrame	-

14.4 Semantics

14.4.1 State Variables

track: VideoStreamTrack
transform: string

14.4.2 Environment Variables

N/A

14.4.3 Assumptions

__init__ is called before any other access program

14.4.4 Access Routine Semantics

__init__(track, transform):

- transition: initiated by track and transform, self.track = track, self.transform = transform
- output: out := self
- exception: ~~N/A~~ TransformationError — thrown if the transform type is unrecognized or if the transformation process fails.

recv(self):

- transition: Processes a video frame (frame) received from a track. Depending on the value of self.transform, it applies one of the following transformations:
  - "HPE": Converts the frame by applying the HPE module annotation.
  - "CM": Converts the frame by applying the CM module annotation.
  - If self.transform is set to any other value, the frame is returned without any transformation.

- output: Returns a new VideoFrame object (new_frame) that has undergone the specified transformation, preserving the original frames timing information (timestamps and time base).
- exception: ~~N/A~~ InvalidFrameError — thrown if the frame is null or corrupted; TransformNotAppliedError — thrown if the transformation cannot be applied.

14.4.5 Local Functions

N/A

15 MIS of SFU Server Module

15.1 Module

SfuServer

15.2 Uses

VideoTransformTrack

15.3 Syntax

15.3.1 Exported Constants

N/A

15.3.2 Exported Access Programs

Routine name	In	Out	Exceptions
consumer	request	-	-
broadcast	request	-	-

15.4 Semantics

15.4.1 State Variables

N/A

15.4.2 Environment Variables

relay: MediaRelay
consumer_track: VideoStreamTrack

15.4.3 Assumptions

N/A

15.4.4 Access Routine Semantics

consumer(request):

- transition: Processes a WebRTC connection request. The function performs the following actions:
  - Parses the request to extract session description parameters.
  - Creates a new RTCPeerConnection object.
  - Logs the information about the sent track.
  - Adds a VideoTransformTrack to the peer connection, which includes subscribing to a consumer track and applying a specified video transformation.
  - Sets the remote description of the peer connection based on the received session description.
  - Creates and sets a local description for the peer connection by generating an answer to the received offer.
- output: Returns a web response in JSON format. This response contains the SDP data and the type of the local description set on the peer connection.
- exception: ~~N/A~~ ConnectionSetupError — thrown if the RTCPeerConnection cannot be established; OfferProcessingError — thrown if the offer cannot be processed or if setting the local/remote description fails.

broadcast(request):

- transition: Manages the setup and handling of a WebRTC peer connection for broadcasting.
  - Parses the incoming request to extract the SDP data.
  - Initializes a new RTCPeerConnection.
  - Adds the peer connection to a global set and logs relevant information.
  - Sets up event handlers for different peer connection events:
    1. Connection State Change: Monitors the connection state, logging changes and closing the connection if it fails.
    2. Track Reception: Handles received tracks, particularly video tracks, by setting a global consumer_track for later use, and logs when tracks end.
    3. Processes the received offer by setting it as the remote description of the peer connection.

4. Creates and sets a local description for the peer connection in response to the offer.

- output: Returns a web response in JSON format, containing the SDP data and the type of the local description set on the peer connection.
- exception: ~~N/A~~ ConnectionSetupError — thrown if the RTCPeerConnection cannot be established; OfferProcessingError — thrown if the offer cannot be processed or if setting the local/remote description fails.

15.4.5 Local Functions

N/A

16 MIS of Human Pose Estimation Annotation Module

16.1 Module

HPE

16.2 Uses

Numpy, CV2, OS, Sys, Time, Subprocess, Shutil, Socket

16.3 Syntax

16.3.1 Exported Constants

server_address, HPE_address, K, pose, Rt1, R1, t1, P1, Identity, P2

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_kpts	Image	List	IOError, ValueError
measureJoint	List, List	Tuple	N/A
matchKpts	List	List	N/A
get3D	List, List	List	N/A

16.4 Semantics

16.4.1 State Variables

N/A

16.4.2 Environment Variables

N/A

16.4.3 Assumptions

External libraries are functioning as expected

16.4.4 Access Routine Semantics

get_kpts(img):

- transition: Saves the input image to a designated path and calls OpenPose to generate keypoints, which are then saved to a JSON file.
- output: Returns a list of keypoints extracted from the input image.
- exception: IOError if image saving or reading fails, ValueError if keypoints processing fails.

measureJoint(kpts1, kpts2):

- transition: Computes the length of the spine in each set of keypoints and returns them ordered by length.
- output: Returns a tuple with the first element being the keypoints set with the longer spine.
- exception: ~~N/A~~ KeyPointError — thrown if keypoints are invalid or insufficient to compute the spine length.

matchKpts(mirror_img):

- transition: Reflects the keypoints from the mirror image to match the real image.
- output: Returns the adjusted keypoints for the mirrored image.
- exception: ~~N/A~~ ReflectionError — thrown if keypoints cannot be reflected properly due to incorrect format or data corruption.

get3D(real_kpts, mirror_kpts):

- transition: Uses the keypoints from the real and mirror images to triangulate 3D points.

- output: Returns the 3D coordinates of the keypoints.
- exception: ~~N/A~~ TriangulationError — thrown if 3D triangulation cannot be performed due to invalid or mismatched keypoints.

16.4.5 Local Functions

N/A

17 MIS of Center of Mass Annotation Module

17.1 Module

CM

17.2 Uses

numpy: for numerical computations
params.bodySegParams: for body segmentation parameters
params.cameraParams: for camera parameters

17.3 Syntax

17.3.1 Exported Constants

K, pose, P1, P2, R1, t1, R2, t2 - Camera intrinsic and extrinsic parameters, and projection matrices derived from them.
foot_in_air_thresh - Threshold for determining if a foot is in the air. CoM_foot_thresh - Threshold for determining the supporting foot based on the center of mass.

17.3.2 Exported Access Programs

Name	In	Out	Exceptions
getCoM	points_3D: 3D points array	CoM: Center of Mass point	-
feetStates	CoM: Center of Mass point points_3D: 3D points array	left_foot, right_foot: States of the feet	-

17.4 Semantics

17.4.1 State Variables

N/A

17.4.2 Environment Variables

N/A

17.4.3 Assumptions

The module assumes that body segment parameters and camera calibration data provided by the bodySegParams and cameraParams modules are accurate and reliable.

17.4.4 Access Routine Semantics

getCoM(points_3D):

- transition: Calculates the center of mass based on the 3D points of body joints.
- output: Returns the 3D coordinates of the bodys center of mass.
- exception: ~~N/A~~ CalculationError — thrown if the center of mass cannot be calculated, possibly due to invalid or insufficient 3D points.

feetStates(CoM, points_3D):

- transition: Determines the state of each foot (left and right) based on their position relative to the center of mass and the vertical distance from the ground.
- output: Returns a tuple containing two dictionaries, left_foot and right_foot, each indicating whether the respective foot is on the ground and whether it is supporting body weight.
- exception: ~~N/A~~ StateDeterminationError — thrown if the states of the feet cannot be determined, perhaps due to invalid center of mass or 3D points data.

17.4.5 Local Functions

N/A

18 MIS of STUN Server Module

18.1 Module

STUN

18.2 Uses

N/A

18.3 Syntax

18.3.1 Exported Constants

STUN_SERVER_ADDRESS

18.3.2 Exported Access Programs

N/A

18.4 Semantics

18.4.1 State Variables

N/A

18.4.2 Environment Variables

N/A

18.4.3 Assumptions

The module assumes that a public STUN server is readily available.

18.4.4 Access Routine Semantics

N/A

18.4.5 Local Functions

N/A

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.

19 Appendix

[Extra information if required —SS]