Precondition: A valid id parameter is provided. The database connection is

operational. Method: GetMeeting

Postcondition: Returns the meeting details or an error message (404 or

Precondition: A valid id parameter is provided, optional schedule

Method: UpdateMeeting

Postcondition: Updates the meeting fields or returns an error (404, 400,

Precondition: A valid id parameter is provided. The database connection is

operational.

Method: DeleteMeeting

Postcondition: Deletes the meeting or returns an error (500).

Precondition: The database connection is operational.

Method: ListMeetings

Postcondition: Returns a list of meetings or an error (500).

Precondition: The request body is valid JSON with admin_id and peer_id (both required). The database connection is operational.

Method: CreateMeeting

Postcondition: Creates a new meeting and returns its details or an error (400 or 500)

Postcondition: Returns an instant meeting's details, including a unique

Precondition: None explicitly required; temporary AdminID and PeerID are

generated within the method.

Method: GenerateMeeting



+ MeetingController: MeetingModel

+ GetMeeting(String): HttpResponse

+ UpdateMeeting(String): HttpResponse

+ DeleteMeeting(String): HttpResponse

+ ListMeeting(DbConnection): List

+ CreateMeeting(String, String, Time)

+ GenerateMeeting(): String

Uses MeetingModel

UserController

- UserController: UserModel

+ SignUp(UserModel): HttpResponse

+ Validate(Cookie): Boolean

+ Login(String, String): HttpResponse

+ NewUserController

Precondition: The request body contains

valid user data in the expected format

(e.g., JSON). Method: SignUp

Postcondition: A new user is created and stored in the database.

Precondition: User must exist in

database

Postcondition: User authenticated,

cookie created

Method: Login

Precondition: User must have logged in

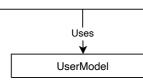
Method: Validate

Postcondition: Cookie Authenticated

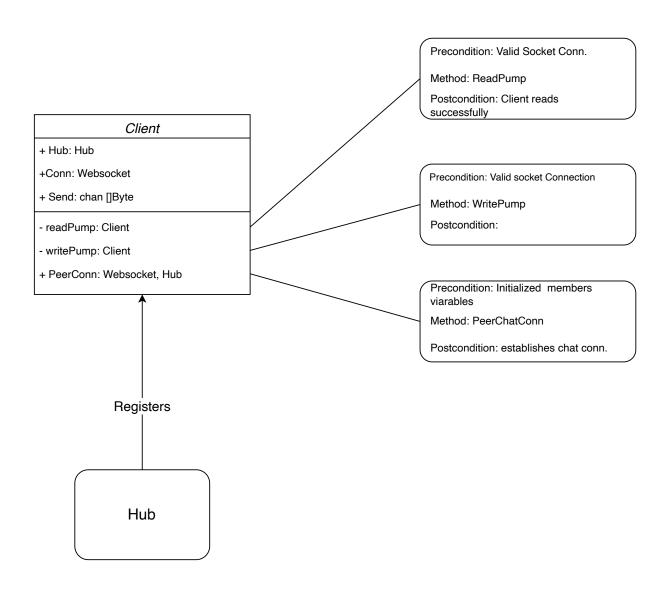
Precondition: Endpoint hit

Method: NewUserController

Postcondition: Exposes controller logic



Chat Package Class Diagram



WEBRTC PACKAGE CLASS DIAGRAM

Room - UUID - Peers - Hub + RoomCreate(UUID): Room + CreateOrGet(UUID): Void

RoomCreate()

Preconditions:

Valid Peer Information: A peer object must be provided.

Hub Association: A hub must be specified as the central entity for room management.

Postconditions:

Room Created: A new room instance is created within the specified hub.

Peer Added to Room: The peer is automatically added to the new room as the first participant.

CreateOrGetRoom()

Preconditions:

Valid Peer Information: A peer object with valid credentials or session data is required.

Hub Specified: A hub the object must be present for room management.

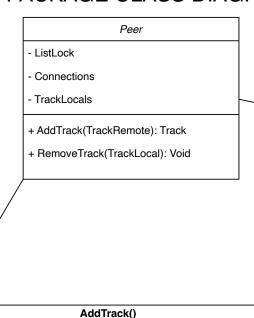
Postconditions:

Room Created or Retrieved:

If the room does not exist, it is created, following the same rules as RoomCreate().

If the room already exists, a reference to the existing room is returned.

WEBRTC PACKAGE CLASS DIAGRAM cont.



Preconditions:

Valid Peer Connection: The peer connection object must be active and valid to allow track addition.

Track Properties: The media track being added (e.g., audio or video) must have proper configurations, such as a codec, source, and unique identifier.

Postconditions:

Track Added to Peer Connection: The track is successfully associated with the peer connection.
Signaling Updated: Necessary signaling messages are sent to other peers to inform them of the new track.

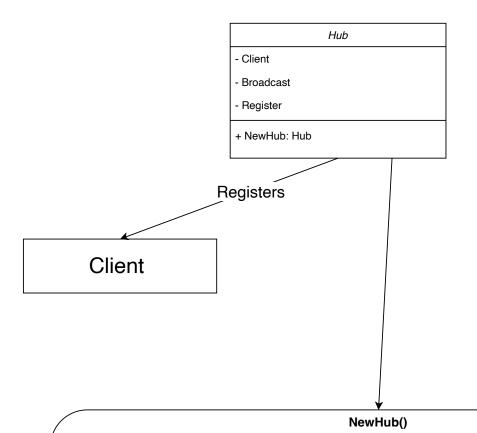
RemoveTrack()

Preconditions: Valid Peer Connection: The peer connection object must be active and valid to allow track removal.

Postconditions:

Track Removed from Peer Connection: The specified track is successfully disassociated from the peer connection.

WEBRTC PACKAGE CLASS DIAGRAM cont.



Preconditions: System Resources Available: The system must have sufficient resources (e.g., memory, network capacity) to create a new hub.

Unique Hub Identifier: A unique identifier for the hub must be provided or generatable to ensure no conflicts with existing hubs.

Postconditions:

Hub Created: A new hub instance is initialized and registered in the system.

Unique Identifier Assigned: The new hub is associated with a globally unique identifier for easy access and management.