WebRTC & Socket.IO Concept Summary

Socket.IO & WebRTC Explained Simply

[OK] What Socket.IO Does in WebRTC

[STEP] Before Connection (Crucial Job):

- [Signal] Helps users find each other
- [Message] Passes the offer, answer, and ICE candidates
- [Secure] Helps setup secure, correct, direct communication

This is the Signaling Phase

[X] After Peer-to-Peer is Connected:

- [X] Socket.IO is no longer involved in video/audio
- [OK] Media flows directly from one user's device to another

This is the WebRTC Connection Phase

[Info] TL;DR Summary:

Phase	Socket.IO Role	WebRTC Role
Signaling phase	[OK] Carries messages	[Pending] Waiting to be configured
Connection phase	[X] Not needed	[OK] Handles media & data stream

[Example] Example:

Shahmeer is calling Ali.

Socket.IO sends a note to Ali:

"Hey, Shahmeer wants to connect. Here's his info."

Ali replies with his own info.

Once both sides agree Socket.IO is done [OK]

WebRTC & Socket.IO Concept Summary

[Pending] After that:

WebRTC handles video/audio calls directly peer-to-peer.

[Info] Socket.IO = the Postman [Postman]

WebRTC = the Phone Call [Call]

[OK] You can still use Socket.IO for:

- Realtime chat (text)
- Room presence (who joined/left)
- Custom control messages (mute, raise hand, etc.)

You now fully understand how signaling + peer connection works in WebRTC.