Real-time WebRTC Multi-Object Detection — Project Documentation

1. Project Overview

- Real-time multi-object detection on live video streamed from a phone to a desktop browser using WebRTC.
- Overlay detections with bounding boxes and labels.
- Collect FPS, latency, and inference metrics.
- Low-resource mode (WASM inference, ~15 FPS capped) for modest laptops.

2. Architecture

Components :

a. Frontend (React + TensorFlow.js WASM)
   - Handles video display, overlays, and FPS/latency metrics.
   - Runs detection locally in browser for low-resource mode.

b. Backend / Signaling Server (Node + PeerJS)
   - Handles WebRTC signaling for phone ↔ desktop connection.
   - Exposes health endpoints and PeerJS server.

c. Docker
   - Provides cross-platform deployment.
   - Avoids .sh limitations on Windows.

d. ngrok
   - Exposes local host to phone allowing accessing without connecting to same network.

Flow:

- Phone connects via QR / ngrok URL → desktop receives WebRTC stream → object detection → overlay → metrics update.

3. Key Features

- Real-time multi-object detection using Coco-SSD.
- FPS, median & P95 latency, and inference time displayed live.
- QR code for easy phone connection.
- Cross-platform deployment using Docker.
- Low-resource mode for laptops without GPU.

4. Limitations

- **WebRTC Connection Drop:** Refreshing either side drops connection; phone must reconnect and re-grant camera permissions.
- **Single Phone Connection:** Only one phone at a time.
- **Metrics:** Benchmarks are simulated; actual values may vary based on network and laptop specs.

5. Design Decisions

- Model Choice: Coco-SSD for lightweight, accurate detection in browser WASM.
- frame thinning to maintain 10–15 FPS.
- **PeerJS Signaling:** Simple and reliable for one-to-one connections.
- **Docker over start.sh:** Ensures cross-platform reproducibility, avoids shell script issues on Windows.

- **Metrics Collection:** Timestamps, frame queue, and inference timestamps to compute E2E latency and FPS.

6. Future Improvements

- Multiple phone streams to one desktop.
- Automatic reconnection on WebRTC drop.
- Server-mode detection (Node/ONNX Runtime) for offloading inference.
- Persistent, real-time metrics in JSON.

7. Usage Notes

- Phone must grant camera access.
- Refreshing either phone or desktop requires reconnection.
- ngrok recommended if phone cannot access desktop directly.
- For implementation refer to readme.md