AI-Powered Video Interview System

Prototype Design & Implementation Guide

---

1. Minimum Inputs Required

From the Hiring Manager / Admin (Pre-Interview Setup)

Input	Purpose	Example Values
Topic	Scope the question bank	`DSA`, `React`, `System Design`, `Backend APIs`
Role Level	Calibrate difficulty	`Intern`, `Junior`, `Mid`, `Senior`, `Staff`
Duration	Control interview length	`15min`, `30min`, `45min`
Focus Areas	Weight specific subtopics	`["arrays", "trees", "dynamic programming"]`
Evaluation Rubric	What to score on	`["correctness", "communication", "edge cases"]`

From the Candidate (At Join Time)

Input	Purpose
Name	Personalization
Resume/Profile (optional)	Tailor questions to experience
Consent	Recording, AI interviewer disclosure

System-Generated Defaults

- Question difficulty progression (easy → medium → hard)
- Time allocation per question
- Follow-up depth limits (2-3 levels max)

---

2. Core System Modules

Module Architecture Overview

```
┌─────────────────────────────────────────────────────────────────┐
│                    FRONTEND LAYER                               │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────────────┐   │
│  │ Video Call   │  │ Code Editor  │  │ Candidate Dashboard  │   │
│  │ (WebRTC)     │  │ (Monaco)     │  │ (Waiting Room, etc.) │   │
│  └──────────────┘  └──────────────┘  └──────────────────────┘   │
└─────────────────────────────────────────────────────────────────┘
                │
                ▼
```

```
┌─────────────────────────────────────────────────────────────┐
│                  ORCHESTRATION LAYER                        │
│  ┌───────────────────────────────────────────────────────┐  │
│  │              Interview State Machine                  │  │
│  │  (greeting → question → follow-up → evaluation → next)│  │
│  └───────────────────────────────────────────────────────┘  │
│                      │                                      │
│       ┌──────────────┼──────────────┐                       │
│       ▼              ▼              ▼                       │
│  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐            │
│  │ Topic/QBank │ │ LLM Engine  │ │ Timer/Flow  │            │
│  │   Loader    │ │  (Claude)   │ │ Controller  │            │
│  └─────────────┘ └─────────────┘ └─────────────┘            │
└─────────────────────────────────────────────────────────────┘
                     │
                     ▼
┌─────────────────────────────────────────────────────────────┐
│                  I/O PROCESSING LAYER                       │
│  ┌─────────────┐ ┌─────────────┐ ┌─────────────────────────┐│
│  │    STT      │ │    TTS      │ │  Video Frame Analyzer   ││
│  │  (Whisper/  │ │ (ElevenLabs/│ │  (Gaze, Tab-switch,     ││
│  │  Deepgram)  │ │   OpenAI)   │ │   Person detection)     ││
│  └─────────────┘ └─────────────┘ └─────────────────────────┘│
└─────────────────────────────────────────────────────────────┘
                     │
                     ▼
┌─────────────────────────────────────────────────────────────┐
│                  DATA & STORAGE                             │
│  ┌─────────────┐ ┌─────────────┐ ┌─────────────────────────┐│
│  │ Session DB  │ │ Transcript  │ │  Evaluation Report      ││
│  │  (state,    │ │   Store     │ │  Generator              ││
│  │  events)    │ │             │ │                         ││
│  └─────────────┘ └─────────────┘ └─────────────────────────┘│
└─────────────────────────────────────────────────────────────┘
```

Module Details

2.1 Interviewer LLM Logic

Purpose: Generate questions, understand answers, ask follow-ups, evaluate

Key Components:

- **System Prompt**: Contains interviewer persona, topic context, rubric
- **Conversation Memory**: Rolling context of Q&A so far
- **Tool Calls**: Structured outputs for question type, evaluation scores

- **Guard Rails**: Prevent off-topic drift, maintain professionalism

2.2 Topic & Question Bank Loader

Purpose: Provide structured question templates and evaluation criteria

Structure:

```json
{
  "topic": "DSA",
  "subtopics": [
    {
      "name": "Arrays",
      "questions": [
        {
          "id": "arr_001",
          "difficulty": "easy",
          "stem": "Find two numbers that add up to a target",
          "follow_ups": ["Time complexity?", "Can you optimize space?"],
          "evaluation_hints": ["Hash map approach", "O(n) optimal"],
          "red_flags": ["Nested loops without optimization discussion"]
        }
      ]
    }
  ]
}
```

2.3 Voice Pipeline (STT + TTS)

STT Options (Speech-to-Text):

Service	Latency	Quality	Cost
Whisper API	~2-3s	Excellent	$0.006/min
Deepgram	~300ms	Very Good	$0.0043/min
AssemblyAI	~1s	Very Good	$0.00025/sec
Browser Web Speech API	Real-time	Fair	Free

TTS Options (Text-to-Speech):

Service	Natural?	Latency	Cost
ElevenLabs	Excellent	~1s	$0.30/1k chars
OpenAI TTS	Very Good	~500ms	$0.015/1k chars
Google Cloud TTS	Good	~200ms	$4/1M chars
Browser Speech Synthesis	Robotic	Instant	Free

2.4 Video/Vision Analysis

Capabilities Needed:

- Face detection & tracking
- Gaze direction estimation
- Multiple person detection
- Tab/window switch detection (browser API)
- Screen share content analysis (optional)

2.5 Meeting Integration

Options:

Approach	Complexity	Realism
Custom WebRTC (Daily.co, Twilio)	Medium	High
Zoom SDK Bot	High	Very High
Google Meet API	High	Very High
Simple browser-based (peer.js)	Low	Medium

---

3. Question Generation, Follow-ups & Evaluation

3.1 Question Generation Strategy

Approach 1: Template + LLM Variation

```
Input: Question bank template
LLM Task: Rephrase naturally, adjust for candidate's prior answers
Output: Spoken question with context
```

Approach 2: Fully Dynamic Generation

```
Input: Topic + difficulty + candidate profile
LLM Task: Generate novel question fitting constraints
Risk: Quality variance, potential bias
```

Recommended: Hybrid Approach

- Use curated question bank for core questions (reliability)
- Use LLM for natural phrasing and contextual follow-ups (flexibility)

3.2 Follow-up Logic

python

```
# Pseudocode for follow-up decision
def decide_follow_up(answer, question_context):
    analysis = llm.analyze(answer, rubric=question_context.rubric)

    if analysis.clarity < 0.5:
        return "clarification"  # "Could you elaborate on..."
    elif analysis.correctness < 0.7:
        return "hint"  # "What if we consider edge case X?"
    elif analysis.depth < 0.6:
        return "probe_deeper"  # "How would you optimize this?"
    elif analysis.correctness > 0.9:
        return "challenge"  # "What's the time complexity? Can we do better?"
    else:
        return "move_on"
```

3.3 Evaluation Framework

Real-time Signals:

Signal	Weight	Measurement
Correctness	30%	LLM semantic comparison to expected answer
Problem-solving approach	25%	Did they clarify? Break down? Consider edge cases?
Communication clarity	20%	Structured explanation, thinking aloud
Code quality (if applicable)	15%	Syntax, readability, efficiency
Time management	10%	Reasonable time per question

LLM Evaluation Prompt Pattern:

```
You are evaluating a technical interview answer.

Question: {question}
Expected Key Points: {rubric}
Candidate's Answer: {transcript}

Rate on these dimensions (1-5 scale):
1. Technical Correctness: Did they get the right answer?
2. Approach Quality: Was their problem-solving systematic?
3. Communication: Did they explain their thinking clearly?
4. Edge Cases: Did they consider boundary conditions?

Provide brief justification for each score.
Return as JSON: {correctness: X, approach: X, communication: X, edge_cases: X, notes: "..."}
```

3.4 Authenticity Detection (Answer Quality)

Red Flags for Memorized/Copied Answers:

- Perfect textbook phrasing without hesitation
- Unable to explain follow-ups on their own answer
- Sudden style/vocabulary shifts mid-answer
- Reading cadence in speech (detected via prosody analysis)

Authenticity Probes:

- "Walk me through that again in different words"
- "What made you think of that approach first?"
- "What would happen if we changed [constraint]?"

---

4. Cheating & Integrity Detection

4.1 Video-Based Detection

Behavior	Detection Method	Confidence
Looking away frequently	Gaze tracking (MediaPipe Face Mesh)	Medium
Second person in frame	Face detection count	High
Reading from another screen	Eye movement patterns + gaze direction	Medium
Phone usage	Object detection in frame	Medium

4.2 Browser/System Detection

Behavior	Detection Method	Confidence
Tab switching	Page Visibility API	High
Window blur	`window.onblur` event	High
Copy-paste in code editor	Clipboard API + input event analysis	High
Typing speed anomalies	Keystroke timing analysis	Medium

4.3 Audio-Based Detection

Behavior	Detection Method	Confidence
Background voices	Speaker diarization	Medium
Text-to-speech playback	Audio fingerprinting	Low
Reading aloud	Prosody analysis	Low-Medium

4.4 Implementation for Prototype

Minimum Viable Integrity Checks:

javascript

```
// Browser-based (easy to implement)
document.addEventListener('visibilitychange', () => {
  if (document.hidden) {
    logEvent('TAB_SWITCH', { timestamp: Date.now() });
    showWarning('Please keep this tab focused');
  }
});

window.addEventListener('blur', () => {
  logEvent('WINDOW_BLUR', { timestamp: Date.now() });
});

// Copy-paste detection in code editor
editor.onDidPaste((e) => {
  const pastedLength = e.range.endColumn - e.range.startColumn;
  if (pastedLength > 50) {
    logEvent('LARGE_PASTE', { length: pastedLength });
  }
});
```

Video Analysis (use MediaPipe):

javascript

```
// Simplified gaze tracking
const faceMesh = new FaceMesh({ locateFile: (file) => `...` });
faceMesh.onResults((results) => {
  if (results.multiFaceLandmarks.length > 1) {
    logEvent('MULTIPLE_FACES');
  }
  // Gaze direction from eye landmarks
  const gazeDirection = calculateGaze(results.multiFaceLandmarks[0]);
  if (gazeDirection.isLookingAway) {
    logEvent('GAZE_AWAY', { direction: gazeDirection });
  }
});
```

---

5. One-Day Prototype Architecture

5.1 Tech Stack (Optimized for Speed)

Layer	Technology	Why
Frontend	Next.js / React	Fast setup, good DX

Layer	Technology	Why
Video	Daily.co or simple WebRTC	Pre-built UI components
Code Editor	Monaco (VS Code editor)	Easy embed, syntax highlighting
STT	Deepgram or Browser API	Low latency, easy integration
TTS	OpenAI TTS or ElevenLabs	Natural voice, simple API
LLM	Claude API	Strong reasoning, good at roleplay
Backend	Node.js / FastAPI	Quick to build
Storage	In-memory / SQLite	No setup overhead

5.2 Simplified Flow

```
┌─────────────────────────────────────────────────────────────┐
│                    PROTOTYPE FLOW                           │
└─────────────────────────────────────────────────────────────┘

[Candidate Joins] → [Consent Screen] → [Waiting Room]
                                              │
                                              ▼
                                    [AI Introduces Itself]
                                              │
                                              ▼
                         ┌──────────────────────────────────┐
                         │   INTERVIEW LOOP                 │
                         │  ┌────────────────────────────┐  │
                         │  │ 1. AI asks question        │  │
                         │  │    (TTS plays audio)       │  │
                         │  └─────────────┬──────────────┘  │
                         │                ▼                 │
                         │  ┌────────────────────────────┐  │
                         │  │ 2. Candidate responds      │  │
                         │  │    (STT transcribes)       │  │
                         │  └─────────────┬──────────────┘  │
                         │                ▼                 │
                         │  ┌────────────────────────────┐  │
                         │  │ 3. LLM evaluates +         │  │
                         │  │    decides next action     │  │
                         │  └─────────────┬──────────────┘  │
                         │                ▼                 │
                         │     [Follow-up OR Next Q]        │
                         └──────────────────────────────────┘
                                              │
                                              ▼
                                     [Closing Remarks]
                                              │
```

```
                    ▼
              [Generate Report]
```

5.3 API Structure

```
POST /api/interview/start
  Body: { candidateName, topic, difficulty, duration }
  Returns: { sessionId, firstQuestion }

POST /api/interview/respond
  Body: { sessionId, transcript, audioBlob? }
  Returns: { evaluation, nextAction, nextQuestion?, audioUrl }

POST /api/interview/end
  Body: { sessionId }
  Returns: { report, scores, transcript }

GET /api/interview/status/:sessionId
  Returns: { currentState, timeRemaining, questionsAsked }
```

---

6. Build vs. Fake Decision Matrix

What to BUILD (Core Demo Value)

Component	Why Build	Effort
Basic video call UI	Visual anchor of the demo	2-3 hrs
LLM conversation flow	Core differentiator	3-4 hrs
Voice input (STT)	"Wow factor" of talking to AI	1-2 hrs
Voice output (TTS)	Human-like interaction	1-2 hrs
Simple question bank	Shows domain knowledge	1 hr
Tab-switch detection	Easy win for integrity	30 min

What to FAKE/MOCK (Time Savers)

Component	How to Fake	Why It's OK
Video meeting bot joining	Hardcode AI avatar, no real "join"	Complex, not core value
Real-time video analysis	Show pre-recorded demo clip	MediaPipe setup is time-consuming
Multi-face detection	Log events, show in admin panel later	Real-time alerting is complex
Full evaluation report	Template with some dynamic fields	Report generation is polish
Code execution	Syntax check only, no actual run	Sandboxing is complex
Persistent database	In-memory, reset on restart	Storage is infrastructure work

Demo Script Optimization

Golden Path Demo (5 minutes):

1. Show admin setting up interview (topic: DSA, level: Junior)
2. Candidate "joins" → sees AI avatar + consent
3. AI greets candidate by name (TTS)
4. AI asks first question (Two Sum)
5. Candidate answers verbally (STT transcribes in real-time)
6. AI asks intelligent follow-up ("What's the time complexity?")
7. Candidate switches tab → Warning appears → Log shown
8. Interview ends → Basic report shown

What to Rehearse:

- Have scripted "good" answers ready
- Know what triggers follow-ups vs. moving on
- Prepare one "cheating" scenario to demo

---

7. Risks, Edge Cases & Limitations

7.1 Technical Risks

Risk	Impact	Mitigation
STT latency/errors	Broken conversation flow	Use Deepgram (fast), show transcript for verification
TTS sounds robotic	Breaks immersion	Use ElevenLabs, or pre-record key phrases
LLM hallucination	Asks nonsensical follow-up	Constrain with structured prompts, use question bank
WebRTC failures	No video/audio	Have fallback text-only mode
Long LLM response time	Awkward silence	Add "thinking" animation, stream response

7.2 Edge Cases

Scenario	How to Handle
Candidate says "I don't know"	Offer a hint, then move on after 2nd "I don't know"
Candidate asks AI a question	Answer briefly, redirect to interview
Candidate goes off-topic	Politely redirect: "Interesting, but let's focus on..."
Background noise issues	Ask to repeat, rely on transcript confirmation
Candidate silent for 30+ sec	Prompt: "Take your time, or would you like me to rephrase?"
Candidate speaks non-English	Detect language, gracefully end or switch

7.3 Limitations to Acknowledge

In Presentation:

- "This prototype handles happy-path scenarios"
- "Production would need robust error handling"
- "Cheating detection is indicative, not forensic"
- "Evaluation is AI-assisted, not AI-decided (human review needed)"

Honest Constraints:

- Can't detect sophisticated cheating (hidden earpiece, second monitor)
- LLM evaluation has ~80% alignment with human evaluators (cite research)
- Voice quality varies with candidate's mic/internet
- Cultural/accent bias in STT is a known issue

---

8. Making the Demo Impactful

8.1 Narrative Structure

Open with the Problem (30 sec):

> "Hiring managers spend 10+ hours/week on screening interviews.
> 60% of candidates no-show or are clearly unqualified within 5 minutes.
> What if AI could handle the first filter?"

Show the Solution (4 min):

- Live demo with golden path
- Show one "cheating" scenario
- Display generated report

Close with Vision (30 sec):

> "This is a prototype. In production, imagine:
> - 24/7 availability across time zones
> - Consistent evaluation across all candidates
> - Human interviewers focus on final rounds only"

8.2 Visual Polish (Quick Wins)

Element	Time	Impact
AI avatar (animated face or Lottie)	1 hr	High

Element	Time	Impact
Branded waiting room	30 min	Medium
Live transcript sidebar	1 hr	High
Progress indicator (Question 2/5)	30 min	Medium
"Recording" indicator	15 min	Low but professional

8.3 Talking Points for Q&A

"How accurate is the evaluation?"

> "We use rubric-based LLM scoring aligned with human interviewer criteria.
> Studies show 75-85% correlation with human scores. This is a filter, not a final decision."

"Can't candidates just cheat?"

> "We detect common cheating: tab switches, gaze away, multiple faces.
> More sophisticated proctoring is a roadmap item.
> Key insight: AI interviews are actually harder to cheat in because follow-ups are dynamic."

"What about bias?"

> "LLM evaluation focuses on technical content, not presentation style.
> We can audit prompts for bias, which is harder with human interviewers.
> Anonymization features (hide name/face during eval) are planned."

"Why would a candidate want this?"

> "Flexible scheduling, no interviewer mood variance,
> instant feedback, less intimidating for some candidates."

Appendix: Sample LLM Prompts

Interviewer System Prompt

```
You are an AI technical interviewer conducting a {topic} interview for a {level} {role} position.

## Your Persona
- Professional but warm
- Patient with pauses and hesitation
- Ask clarifying questions when answers are ambiguous
- Never give away answers directly

## Interview Structure
- You have {n} questions to cover in {duration} minutes
- Start with easier questions, progress to harder
- For each question: ask → listen → follow-up (max 2) → evaluate → next
```

```
## Current Question
{question_json}

## Evaluation Criteria
{rubric}

## Conversation History
{history}

## Your Task
Based on the candidate's last response, decide:
1. Ask a follow-up (if answer was partial/unclear)
2. Move to next question (if answer was complete or 2 follow-ups done)
3. Provide a brief acknowledgment before the next question

Respond in JSON:
{
  "acknowledgment": "Brief response to their answer (1-2 sentences)",
  "action": "follow_up" | "next_question" | "end_interview",
  "next_utterance": "What you'll say next",
  "evaluation": {
    "correctness": 1-5,
    "communication": 1-5,
    "notes": "Brief observation"
  }
}
```

Follow-up Generation Prompt

```
The candidate was asked: "{question}"
They answered: "{answer}"

The expected key points were: {expected_points}

Generate ONE concise follow-up question that:
- Probes deeper if they got it right ("How would you optimize?")
- Offers a hint if they're stuck ("What if we used a hash map?")
- Clarifies if ambiguous ("When you say X, do you mean...?")


Keep it under 20 words. Be conversational.
```