Andrew: A Digital Platform for AI-Based Virtual Interview Assistant

Umar Farooq Azmi[1], Rakesh Choudhary[2], Shivam Lohar[3], Moajez Lalani[4], Dr. Shikha Gupta[5]
[1,2,3,4,5] *Department of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, India*

***Abstract*—This study presents Andrew, an artificial intelligence- driven virtual assistant built to conduct domain-specific technical as- sessments across roles including Java Developer, Full-Stack Developer, Data Structures & Algorithms, DevOps Engineer, Frontend Developer, and Backend Developer. Moving beyond standard mock interview appli- cations, our platform measures candidate performance comprehensively by evaluating several distinct metrics, specifically answer quality, domain expertise, communication clarity, confidence levels, and problem-solving capabilities. The platform captures these metrics via computer vision-based proctoring (MediaPipe FaceMesh), audio analysis (faster-whisper speech recognition), alongside natural language processing capabilities (a fine-tuned Phi-3-mini utilizing QLoRA). We integrated a real-time behavioral integrity component to monitor gaze shifting, blink frequency, head positioning, and facial presence, allowing the software to calculate a proctoring score parallel to the main interview assessment. Following the session, the user receives an extensive feedback document featuring radar chart visualizations, a question-by-question performance breakdown, and detailed behavioral analytics. The entire architecture executes locally on standard consumer-grade hardware (RTX 2050, 4 GB VRAM) without any cloud dependencies during active sessions, making structured, data- informed interview practice widely accessible.**

***Index Terms*—AI, behavioral proctoring, computer vision, fine- tuning, natural language processing, QLoRA, speech recognition, virtual interview assistant.**

I. INTRODUCTION

Campus placement preparation at many engineering col- leges is still a scattered, high-stress experience. Candidates usually have to hunt down practice resources across dozens of websites, relying heavily on the availability of seniors, professors, or overwhelmed placement cells to conduct mock interviews. What ought to be a highly structured learning phase devolves into an uncertain time sink.

Countless students pour hours into rote memorization. Yet, they receive almost zero concrete feedback on crucial metrics like body language, confidence levels, or the actual quality of their vocal delivery. High-quality guidance lives scattered across informal channels, YouTube videos, and random blogs. Because they lack continuous, data-driven practice, otherwise capable candidates frequently miss out on top opportunities simply because they do not know where they are failing.

The surge in AI-focused hiring has shifted industry expecta- tions. Recruiters now demand a tight combination of technical sharpness—especially in Machine Learning and Generative AI—paired with highly polished soft skills. Standard univer- sity curricula rarely address this dual requirement effectively.

Project Andrew directly tackles these widespread prepara- tion gaps. We engineered it as a completely voice-enabled artificial intelligence interviewer built to run entirely on acces- sible, consumer-grade hardware (specifically, an RTX 2050, 4 GB VRAM, 24 GB RAM, running Windows 11 and Python 3.12.5). Standard text-based mock interview chatbots feel artificial. Andrew, by contrast, executes the entire session through real-time vocal exchanges. The system captures the user's speech and transcribes it locally using faster-whisper. A fine-tuned Phi-3-mini model (andrew_v2_Q4_K_M.gguf) processes that input, immediately returning a response via Piper TTS audio to simulate a genuinely high-pressure, re-alistic interview scenario. While the conversation happens, a concurrent MediaPipe FaceMesh proctoring layer actively monitors the webcam feed. It tracks eight specific violation types in real time to generate a behavioral integrity score, which is then paired with a GPT-4o-mini semantic evaluation.

II. LITERATURE REVIEW

A. AI-BASED INTERVIEW CRITIQUE SYSTEMS

An interview preparation companion utilizing deep learning was introduced as an AI-based critique system in [4]. By integrating NLP to analyze spoken answers and computer vi- sion to track physical gestures, the platform effectively shifted from static assessments to dynamic, data-driven feedback. Yet, real-time multi-modal integration still struggled with practical limitations. The synchronization between the semantic content analysis and the incoming behavioral signals simply lacked consistent robustness during live execution.

B. DEVELOPMENT OF AI-BASED INTERVIEW SYSTEMS

Remote hiring support drove the design of the AI interview system detailed in [5], leveraging large-scale analytics along- side non-verbal signals like facial expressions and eye move- ments. While this setup proved highly effective for filtering candidates on the recruitment side, it simply wasn't built for student training. It ultimately lacked the formative feedback loops necessary to drive continuous skill improvement.

C. AI MOCK-INTERVIEW PLATFORMS

Candidate-performance indicators, particularly confidence- oriented metrics, successfully enhanced interview practice within the platforms evaluated in [3] and [6]. Yet, these setups often relied on highly generic questioning patterns.

They struggled to generate the deep, domain-aware technical questions required for specific industry roles.

D. OPENFACE FOR FACIAL BEHAVIOR ANALYSIS

Frameworks like OpenFace [1] excel at extracting precise landmarks, head-pose metrics, and gaze tracking. While they form the essential foundation for behavioral analysis, they fall short of operating as standalone interview platforms. Without external language modules, they completely miss the conversational context and semantic meaning of a candidate's response. Andrew bridges this exact gap. It merges the visual proctoring capabilities of MediaPipe FaceMesh directly with GGUF-based language model inference to create a single, unified pipeline.

E. BERT FOR LANGUAGE UNDERSTANDING

Transformer architectures like BERT [2] drove massive improvements in contextual language understanding and se- mantic similarity. Within interview platforms, these models are excellent at evaluating whether a candidate's answer is actually relevant. The problem is that relying solely on text completely ignores delivery dynamics. Crucial elements like tone, pauses, and overall confidence are lost entirely. Andrew solves this limitation. It pairs the semantic evaluation of GPT-4o-mini with a dedicated behavioral integrity module to independently quantify those exact delivery signals.

F. SMART PREP AND MULTI-AGENT INTERACTIVE SYS- TEMS

Multi-agent pipelines like the Smart Prep architecture have clearly advanced conversational naturalness alongside overall user interaction quality [7], [8], [10]. Yet, processing video, text, and audio simultaneously in real time continues to trigger significant coordination and latency hurdles for these systems. Andrew bypasses this bottleneck. By completely isolating the active proctoring loop (processing 150 ms FaceMesh frames using setTimeout) from the voice pipeline, the system suc-cessfully prevents any cross-contamination of latency budgets.

III. PROBLEM STATEMENT

Even with AI tools readily available, countless students still default to manual preparation and practice without any con- tinuous, structured guidance. Repeated peer-level observations highlight five recurring hurdles:

1) Candidates over-focus on technical content, completely neglecting soft skills and communication.
2) High-quality coaching is highly inconsistent and rarely personalized to the individual.
3) Practice sessions remain entirely manual and completely disconnected from data-driven metrics.
4) Current web platforms suffer from cloud dependency, severely limiting their real-time

reliability.

5) A massive number of students simply lack access to an integrated, practical tool for realistic preparation.

A large segment of learners remains completely underserved simply because a unified, easy-to-use platform does not exist. Project Andrew directly tackles these exact five deficiencies.

IV. EXISTING SYSTEM

Traditional, scattered interview preparation remains the norm across many engineering institutions. Candidates lean heavily on coding platforms, repositories of common ques- tions, placement cells, and the availability of professors or seniors. This method certainly improves basic technical recall. Yet, it completely fails to train students for the reality of live interview communication.

Plenty of online mock interview tools exist today. The problem is their fundamentally static design. Most simply feed users pre-recorded questions or offer rigid chatbot interactions that cannot adapt to the conversation. Post-interview feedback usually stops at basic correctness checks and generic advice, completely ignoring crucial factors like answer structuring, confidence, and how well a candidate handles follow-up pres- sure.

Heavy reliance on cloud processing plagues many modern AI-based systems. This architecture drives up operational costs and severely restricts sustained, regular practice access for users. Without reliable access, students rarely experience a genuinely high-pressure simulation before their actual inter- view day. Andrew bypasses these exact limitations. The entire active interview pipeline runs locally, only calling GPT-4o- mini for the final post-interview evaluation rather than pinging the cloud during the live session.

V. PROPOSED SYSTEM

Andrew integrates technical and behavioral interview eval- uation into a single platform, transforming manual and sub- jective mock interview practice into a measurable digital workflow. The complete technology stack is summarized in Table I.

A. USER INTERFACE (FRONTEND)

The frontend is developed using React 18 with Vite and Tailwind CSS. It consists of five pages routed via React Router v6, with global state managed in App.jsx:

- Landing.jsx: Split layout with marketing copy and JWT-authenticated sign-in/sign-up tabs.
- Dashboard.jsx: Four stat cards (total interviews, average overall score, best score, improvement delta), a Recharts line chart for score progression, a six-role selector, and a Begin Interview button.
- Interview.jsx: Google Meet-style layout with a camera tile (left) and Andrew avatar (right). Implements MediaRecorder audio capture, 2-second silence detection, Whisper transcription, GGUF response generation, Piper TTS playback, and a 30-minute countdown timer. Camera tile overlays show focus score bar, gaze dot, eye contact percentage, and coaching warnings.
- Results.jsx: Two-column layout (lg breakpoint) with radar chart, multi-dimensional scores, strengths and weaknesses, per-question breakdown tab, transcript tab, and behavioral analysis section including integrity verdict badge, violation grid, and focus timeline chart.

Table I FULL TECHNOLOGY STACK

Layer	Technology	Purpose
AI Model	andrew_v2_Q4_K_M.gguf	Interview persona and question generation
LLM Runtime	llama-cpp-python	Runs GGUF model locally (20 GPU layers)
Evaluation	GPT-4o-mini	Semantic scoring and behavioral analysis
Backend	FastAPI + Uvicorn	REST API server
Frontend	React + Vite + Tailwind	Web UI

STT	faster-whisper small	Speech to text (CPU, int8)
TTS	Piper TTS en_US-ryan-high	Text to speech (WAV streaming)
Proctoring	MediaPipe FaceMesh	Facial landmark analysis
Database	SQLite + SQLAlchemy	Persistent session storage
Auth	JWT + bcrypt	User accounts and sessions
Charts	Recharts	Dashboard progress visualization

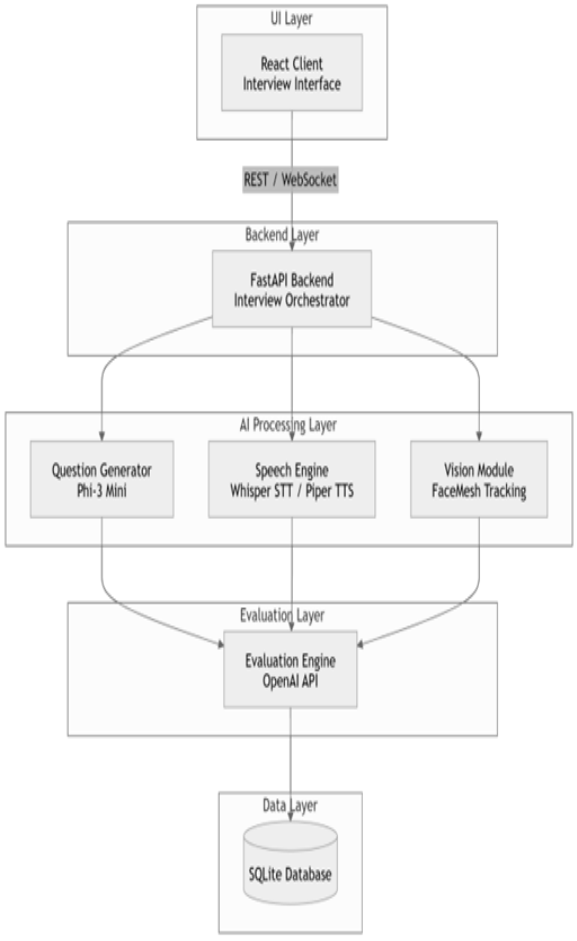

Fig. 1. System Architecture

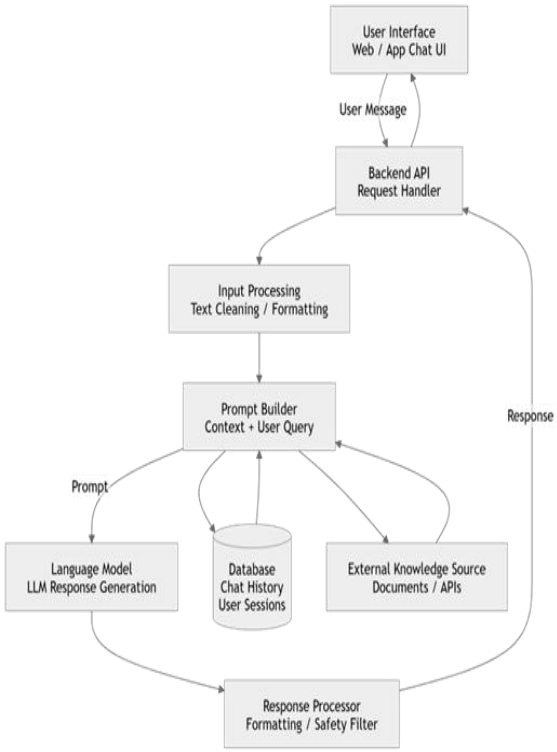

Fig. 2. Chatbot Architecture

- History.jsx: Past interviews listed newest first with expandable cards containing scores, behavior, breakdown, and transcript tabs. Supports deletion with confirmation dialog.

B. SPEECH PROCESSING MODULE

The browser captures incoming audio natively through the MediaRecorder API. It relies on the WebM/Opus codec combined with a strict 100 ms timeslice (recorder.start(100)). Forcing this specific timeslice guarantees that the collected chunks successfully generate data blobs exceeding 25,000 bytes. We then pass that audio blob through ffmpeg for a standard WebM to WAV conversion. From there, faster-whisper (specifically the small, int8 model executing directly on the CPU) handles the entire local transcription process. By running this exact pipeline, the platform completely eliminates cloud dependency during an active live session. It also effortlessly maintains sub-second transcription latency across every single conversational turn.

C. AI INTERVIEW ENGINE

At the core of the interview engine sits andrew_v2_Q4_K_M.gguf. We fine-tuned this

Phi- 3-mini model using QLoRA ($r = 16$, NF4 4-bit quantization). Execution relies on llama-cpp-python. It offloads 20 GPU layers directly to the RTX 2050, consuming approximately 2.3 GB of VRAM while maintaining a 2048-token context window. By passing the entire messages array—representing the complete conversation history—with every single call, the system strictly enforces contextual follow-ups throughout the interview.

A strict four-phase architecture controls the difficulty of the questions. Before triggering an inference call, the system in- jects a programmatic nudge that explicitly outlines the current phase alongside recently discussed topics. This mechanism drives progressive difficulty and topic diversity, ensuring the platform never relies solely on the limited memory constraints

Table Ii BEHAVIORAL VIOLATION TYPES

Violation Type	Detection Method	Threshold
looking_away	Head YAW via ear landmarks	> 0.38
looking_down	Head PITCH via nose forehead	> 0.38
looking_up	Head PITCH negative	< –0.30
no_face	No landmarks detected	5 s absence
suspicious_gaze	Iris offset from eye center	> 0.045
low_blink	EAR in 30 s window	< 4 blinks
high_blink	EAR in 30 s window	> 40 blinks
mouth_open	Mouth-open ratio sustained	20+ frames

of a small model. Before delivering the final output to the user, a dedicated hallucination guard actively strips out unwanted teaching-style phrases. Once the session reaches six or more questions, a lightweight GPT-4o-mini call seamlessly handles wrap-up detection. Finally, the system selects one of four randomized closing phrases to immediately trigger the evaluation pipeline.

D. BEHAVIORAL INTEGRITY MODULE (useBehaviorAnalysis.js)

MediaPipe FaceMesh is loaded from CDN (not npm, due to Vite incompatibility) and processes webcam frames every 150 ms via a setTimeout loop with a 3-second WASM warm-up delay. Eight violation types are tracked as detailed in Table II.

VI. EVALUATION PIPELINE

A. VOICE OUTPUT MODULE

To drive the voice output, Piper TTS (en_US-ryan-high.onnx) translates Andrew's text responses into a 22050 Hz, 16-bit PCM WAV format. The system then streams this buffer directly to the browser as a Blob URL. During development, we encountered a specific bug where header finalisation behaviour caused Piper to output a 44-byte empty WAV file. Calling wav_file.close() directly before the buffer read resolved the issue entirely, ensuring the generation of valid outputs well over 112,000 bytes. A second playback glitch caused audio.play() to continuously interrupt itself. We eliminated this by removing the manual play() calls completely. The audio solution now relies entirely on the onloadedmetadata event combined with an srcObject assignment.

B. SEMANTIC EVALUATION MODULE

GPT-4o-mini handles the entire post-interview evalua- tion. The system first builds the full transcript directly from the session messages array. It then merges this text with a specific behavior context string, which is cal- culated using the FaceMesh integrity metrics and viola- tion counts. We use a single, highly structured prompt to request a detailed JSON payload. This response must include several specific fields: technical_score, domain_coverage, communication, confidence, and problem_solving (all scored from 0–10). It also demands qualitative outputs like strengths, weaknesses,
a final_comment, and a decisive verdict (Strong Hire / Hire / Maybe / No Hire / Strong No Hire). Finally, the JSON extracts the integrity_score, integrity_verdict, and behavior_notes. Before parsing this JSON, the back- end actively strips away any Markdown code fences. The parsed result is then saved persistently to the SQLite interviews table and pushed straight to the frontend for immediate user display.

C. DATABASE SCHEMA

The SQLite database (andrew.db) is managed by SQLAlchemy with two tables. The users table stores UUID primary key, name, email, bcrypt password hash, and creation timestamp. The interviews table stores UUID primary key, user_id foreign key, role, verdict, five numeric scores, strengths, weaknesses, final_comment, questions_asked, transcript (JSON string),

per_question breakdown (JSON string), integrity_score, integrity_verdict, behavior_notes, and creation timestamp.

D. COMPLETE API REFERENCE

The complete REST API reference is provided in Table III.

E. SYSTEM FLOW

The complete end-to-end data flow proceeds as follows:

1) User selects a role on the Dashboard and clicks Begin In- terview. Dashboard sends POST /start_interview and stores the returned session_id and open- ing question in global state before navigating to interview.
2) useBehaviorAnalysis initialises MediaPipe FaceMesh after a 3-second WASM warm-up. Andrew's opening question is spoken via Piper TTS.
3) Candidate records an answer via MediaRecorder. Silence detection fires after 2 seconds of quiet, sending the audio blob to POST /transcribe.
4) The Whisper transcript is forwarded to POST/answer. The backend appends to the messages array, injects a phase nudge, checks for wrap-up via GPT-4o-mini (after 6+ questions), runs GGUF inference, strips hallucinations, and returns Andrew's next question.
5) Andrew speaks the response via Piper TTS. Steps 3–5 repeat until wrap-up is triggered or the candidate clicks End.
6) getBehaviorSummary() captures all FaceMesh metrics. stopCamera() tears down the webcam. POST /evaluate sends the full transcript and behav- ior summary to GPT-4o-mini.
7) The evaluation JSON is saved to SQLite and returned to the frontend. onFinish() sets global evaluationData state and navigate('/results') renders the results page.

Table III Rest Api Endpoints

Method	Endpoint	Auth	Request Body	Response
POST	/auth/signup	No	name, email, password	token, user
POST	/auth/login	No	email, password	token, user
GET	/auth/me	Yes	–	user object
POST	/start_interview	No	candidate name, role, user id	session id, andrew
POST	/answer	No	session id, answer	andrew, question number, wrap up
POST	/transcribe	No	audio file (multipart)	transcript
POST	/speak	No	session id, answer (text)	WAV stream
POST	/evaluate	No	session id, answer, behavior	full evaluation object
GET	/history	Yes	–	interviews array
GET	/progress	Yes	–	progress array, stats
DELETE	/interview/{id}	Yes	–	{success: true}

VII. IMPLEMENTATION CHALLENGES AND RESOLUTIONS

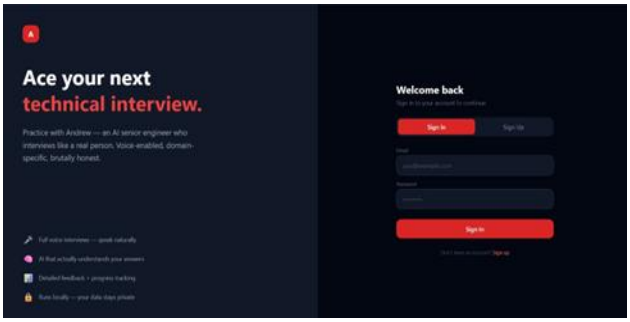

Fig. 3. Login Page

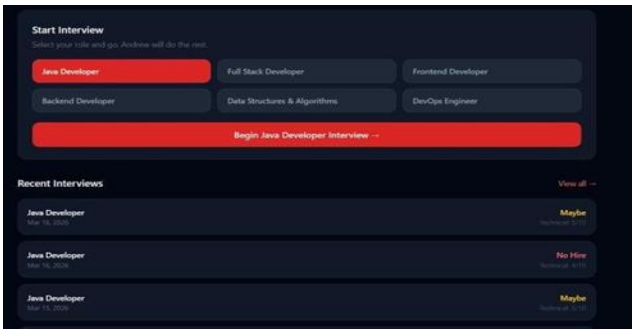

Fig. 4. Dashboard

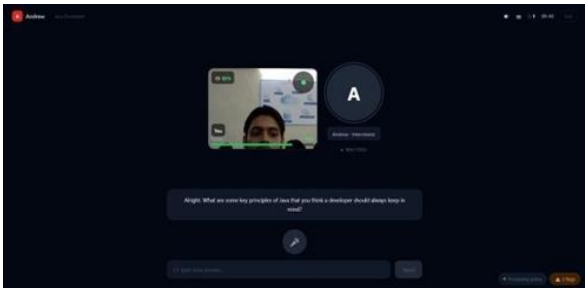

Fig. 5. Starting Interview

Fig. 6. Conducting Interview

Fig. 7. Results Breakdown

Fig. 8. Results Overview

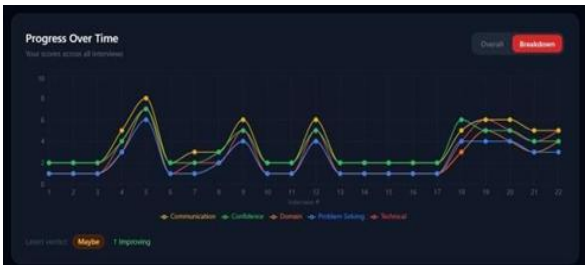

Fig. 9. Progress Breakdown

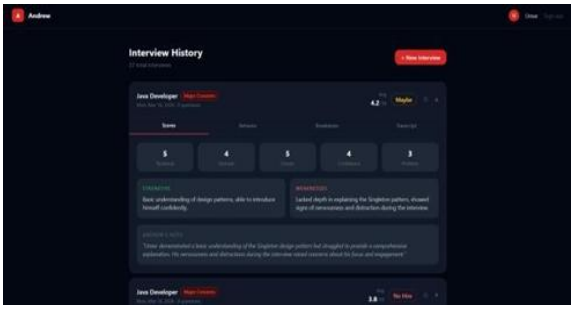

Fig. 10. Candidate History

Fig. 11. Transcript History

A. MODEL TRAINING

The interviewer model was developed through a two-phase fine-tuning process. In Phase 1, Phi-3-mini-4k-instruct (3.8 B parameters) was fine-tuned using QLoRA ($r = 16$, NF4 4-bit quantisation, 1 epoch) on 2,679 domain-specific Q&A entries across three epochs, reducing training loss from 1.65 to 0.56 in 3 hours 29 minutes on an RTX 2050 GPU. This phase produced answer-centric behaviour.

In Phase 2, 10,000 interviewer-style conversations were distilled using GPT-4o-mini across six domains and three difficulty levels. A one-epoch top-up fine-tuning run at a reduced learning rate of 10^{-4} preserved existing knowledge while adding interviewer persona behaviour. The final model, andrew_v2_Q4_K_M. gguf, is 2.28 GB and runs at ap- proximately 2.3 GB VRAM with 20 GPU layers offloaded.

B. QUESTION GENERATION AND GUARDRAILS

Each question is generated using the full conversation history and a programmatic phase nudge injected before inference. The nudge defines the active interview stage (phases 1– 4) and recently covered topics, enabling progressive difficulty and topic diversity without relying exclusively on small-model memory. A hallucination guard post-processes responses to remove teaching-style output before

delivery to the candidate.

Bug	Root Cause	Fix
Audio corruption from Piper TTS	`wav_file.close()` called after `seek(0)`	`close()` before `seek(0)`
Empty audio blobs from MediaRecorder	`recorder.start()` with no timeslice	`recorder.start(100)`
Topic nudge appearing in transcript	Role set to `'user'`	Changed to `'system'`
bcrypt password verify failing	`passlib` incompatible with `bcrypt 5.0.0`	Switched to direct bcrypt
`audio.play()` interrupted	Manual `play()` conflicting with `srcObject`	Removed manual `play()`; used `onloadedmetadata`
FaceMesh not a constructor	MediaPipe npm incompatible with Vite	Load from CDN via script tag
FaceMesh not detecting faces	WASM not loaded before first frame	3 s `setTimeout` before `sendFrames()`
`/history` 500 error	Integrity columns missing from SQLAlchemy model	Added 3 columns to Interview model
Results stuck on loading	`navigate()` called before `onFinish` set state for /results	Moved navigate inside callback
Delete button always 404	`uvicorn --reload` serving stale worker	Restart without `--reload` flag
Everything or nothing flagged	Thresholds too strict then too loose	Tuned to 0.35–0.38 with frame accumulation

C. VOICE PIPELINE

Audio is captured in WebM format, converted to WAV via ffmpeg, and transcribed using faster-whisper. A critical issue was resolved in which MediaRecorder produced only 110-byte output when no timeslice was specified. Using recorder.start(100) corrected chunk collection and increased blob size beyond 25,000 bytes. On the output side, Piper initially produced a 44-byte empty WAV due to header finalisation behaviour; explicit wav_file.close() before buffer read resolved this and produced valid output beyond 112,000 bytes.

D. BEHAVIOURAL PROCTORING PIPELINE

The proctoring layer runs concurrently with the voice loop via a decoupled setTimeout chain. Camera device selection uses keyword matching (logi/c270/usb/webcam) and falls back to the last enumerated device. getUserMedia is called directly, bypassing the Camera utility which ignores deviceId options. Violation thresholds were tuned across multiple test sessions: initial values of 0.40/0.45 flagged every head movement, while loosened values missed genuine vio- lations. Final thresholds (0.35–0.38 for head pose, 0.045 for iris offset) combined with a frame-accumulation requirement before firing produce reliable violation detection without no- tification fatigue.

E. BUGS RESOLVED

Table IV summarises the significant bugs encountered and resolved across development sessions.

VIII. RESULTS AND DISCUSSION

A. OBSERVED FINDINGS

Extensive testing confirmed the efficiency of our voice pipeline. On the target hardware (RTX 2050, 24 GB RAM), every interaction consistently maintained less than one second of latency. The semantic evaluation powered by GPT-4o- mini successfully avoided generic scoring. Instead, it delivered highly calibrated, transcript-specific feedback. It also reli- ably separated genuine conceptual understanding from purely memorized answers. Across multiple testing sessions, domain locking proved highly stable. At the same time, the built- in hallucination guardrails effectively caught and corrected any edge-case teaching responses. Finally, we analyzed the behavioral integrity score. During manual review sessions, these automated metrics closely matched our own subjective assessments of candidate attentiveness.

B. SOCIAL AND IMPACT BENEFITS

- Democratisation of coaching: Top-tier interview coaching becomes completely accessible, regardless of a student's financial situation.
- Confidence building: Consistent, private practice sessions actively strip away the anxiety associated with real-world interviews.
- Improved employability: Candidates receive highly struc- tured feedback. This directly sharpens their communica- tion, deepens domain knowledge, and refines problem- solving skills.
- Self-awareness: In-depth behavioral analytics and com- prehensive reports expose critical blind spots. Students can then focus entirely on targeted, specific improve- ments.
- Accessible anytime: The platform guarantees continu- ous availability directly on standard consumer hardware. Users face absolutely zero per-session cloud fees.

IX. KNOWN LIMITATIONS

Initial startup currently incurs a 15–20 second latency

penalty. This delay happens simply because the system must load Whisper, the Phi-3-mini (GGUF) model, and Piper TTS simultaneously during boot. We plan to introduce a dedicated pre-warm routine to cut down this wait time. Right now, the backend relies on a temporary, in-memory Python dictionary to store active interview sessions. Consequently, any server restart completely wipes that session data. Finally, the user signup form currently lacks basic email format validation. We have flagged all of these specific items for immediate resolution in our upcoming development cycle.

X. FUTURE WORK

1) Adaptive question sequencing: Dynamically increase or decrease question difficulty based on real-time candidate performance trends rather than fixed phase boundaries.
2) Session persistence: Migrate in-memory session storage to Redis or SQLite to survive server restarts.
3) PDF export: Generate downloadable interview reports for portfolio and faculty review.
4) Human-in-the-loop evaluation: Enable faculty review and annotation of AI-generated reports for hybrid feed- back.
5) Extended domain coverage: Expand support to non- technical domains such as MBA interviews and civil services.
6) Bias mitigation: Improve fairness across accents and demographic variation through diversified fine-tuning data.
7) System optimisation: Reduce startup latency through model pre-loading and pipeline parallelisation.
8) Docker containerisation: Package the full stack for one- command deployment on any machine.

XI. CONCLUSION

Building a fully functional, voice-enabled AI technical interviewer does not require massive server budgets. Project Andrew proves that this technology can be successfully de- ployed directly on consumer-grade hardware, keeping recur- ring costs to an absolute minimum. The architecture brings several distinct technologies together. It relies on local GGUF language model inference (specifically a QLoRA fine-tuned Phi-3-mini) alongside real-time voice processing powered by faster-whisper STT and Piper TTS. While this happens, MediaPipe FaceMesh tracks eight distinct violation types for concurrent behavioral proctoring. Finally, GPT-4o-mini handles the post-interview semantic evaluation. By unifying these layers, the system creates a genuinely high-pressure, realistic interview simulation. Crucially, it achieves this with zero cloud dependency during the active live session.

Standard, static question-answer tools simply cannot match this. Our platform actively maintains the complete conversa- tion context. It strictly enforces domain-specific probing by utilizing phase nudges combined with hallucination guards. At the same time, it actively computes a dedicated integrity score derived from facial behavioral signals. The final result is a set of highly calibrated, multi-dimensional verdicts backed entirely by hard transcript evidence. Ultimately, this approach effectively closes the wide gap between simply studying theory and achieving true practical interview readiness. It puts structured, data-driven coaching directly into the hands of the target user population, running seamlessly on hardware they already own.

REFERENCES

[1] S. Prakash and P. Sahu, “Bring AI-Powered Interview Assistant Chat- bot,” International Journal of Scientific Research and Engineering Development, May–Jun. 2024.
[2] S. Kolpe, S. Patil, J. Deshmukh, S. Jeughale, and Y. Misal, “AI Based Mock-Interview Behavioural Recognition Analyst,” vol. 12, no. 5, May 2024.
[3] N. Gupta, A. Sharma, S. Shukla, and P. Kumar, “AI-based Mock Interview Evaluator: An Emotion and Confidence Classifier Model,” vol. 9, no. 3, 2025.
[4] S. V. Nirgide, S. A. Aktharali, P. N. Patil, S. V. Raktate, and M. F. Mushtaque, “AI Based Interview Critique System: Interview Preparation Companion Using Deep Learning,” vol. 12, no. 2, Feb. 2024.
[5] S. G. Benny, A. Prakash, B. Wilson, S. S. Kumar, and N. Philip, “An Interview System Using AI Technology,” vol. 5, no. 2, Mar.–Apr.

2024.

[6] M. R. Madanachitran, A. Austin, K. Balaji, M. Rajappan, and W. Li, "AI Mock Interview Chatbot Using Gen AI," 2024.

[7] A. B. Nofal, H. Ali, M. Hadi, A. Ahmad, A. Johri, and J. Qadir, "AI- Enhanced Interview Simulation in the Metaverse," vol. 8, 2024.

[8] S. Deote, V. Pawar, Y. Pandilwar, S. Chandra, and G. Bawankule, "AIPrepMate: AI-Assisted Mock Interview and Feedback System," vol. 14, no. 3, Mar. 2025.

[9] S. M. Patil, K. V. Shinde, B. G. Vakare, and S. S. Dunbale, "AI Powered Mock Interview Platform," vol. 13, no. 1, Feb. 2025.

[10] S. Pawar, G. Misal, V. Sanap, V. Nanaaware, and N. Berwal, "Empow- ering Interview Success: An AI-Driven Approach," vol. 12, no. 5, May 2024.