Eightfold AI × ArIES Hackathon

CodeSage: The AI Technical Interviewer

Building Intelligence That Evaluates Intelligence

The Challenge: Build an AI technical interviewer that conducts live, adaptive coding interviews with human-like insight. It must analyze code in real-time, provide context-aware hints, assess code quality beyond correctness, and generate detailed performance reports, all while ensuring scalability, consistency, and unbiased evaluation.

1 Problem Context

Technical interviews are the cornerstone of modern hiring pipelines, especially for software engineering roles. Yet, despite their importance, they remain one of the most challenging aspects of the recruitment process to standardize and scale. Traditional interviews rely heavily on the skill, consistency, and availability of human interviewers, all of which introduce friction:

- Bias and Subjectivity – Different interviewers may evaluate the same candidate differently based on unconscious bias, communication style preferences, or varying technical depth.
- Inconsistency and Variability – The difficulty, pacing, and guidance provided in interviews often vary, leading to an uneven experience that doesn't accurately reflect candidate ability.
- Scalability Limitations – Conducting high-quality, interactive technical interviews for hundreds of candidates quickly becomes logistically and financially taxing.
- Incomplete Signal – Many interviews stop at whether the final solution works, ignoring how candidates think, adapt under pressure, or respond to guidance, all of which are crucial signals for predicting on-the-job performance.

2 CodeSage

2.1 Core Requirements

1. Real-time Code Analysis Engine

- Continuously track code as it's written and run it safely in a sandboxed environment.
- Detect syntax and runtime errors instantly, helping candidates correct mistakes quickly.
- Measure execution time, memory usage, and estimate algorithmic complexity (Big O).

2. Agentic Interviewer Hint System

- Adjust problem difficulty dynamically based on the candidate's performance.
- Ask follow-up questions that probe deeper into the candidate's understanding.

- Offer hints progressively (Nudge → Guide → Direction) to help candidates move forward.

3. **Deep Code Quality Assessment**
   - Assess style, readability, and adherence to standards (e.g., PEP 8).
   - Identify the candidate's problem-solving approach and methodology.
   - Recommend alternative solutions or optimizations for better performance.
4. **Comprehensive Reporting System**
   - Present all candidate metrics in a clean, visual dashboard for hiring managers.
   - Generate a human-readable summary of the candidate's performance and problem-solving journey.
   - Provide a session playback feature to review code evolution, hints used, and candidate thought process.
5. **User Interface & Audio Interaction**
   - Provide an intuitive, interactive UI that allows candidates to code, receive hints, and view feedback seamlessly.
   - Integrate audio input to enable candidates to think aloud, explain their approach, or answer follow-up questions verbally.

3 Sample Interaction

Problem: Implement a function to find duplicates in an array.
CodeSage: "Let's start with this problem. Take your time and think aloud as you code."
Candidate: "I'll use nested loops to compare each element."

```
def find_duplicates(arr):
    duplicates = []
    for i in range(len(arr)):
        for j in range(i+1, len(arr)):
            if arr[i] == arr[j]:
                duplicates.append(arr[i])
    return duplicates
```

CodeSage: "Your logic is correct! I notice this has $O(n^2)$ complexity. For a million-element array, that's a trillion operations. Can you think of a way to solve this in a single pass?"
Candidate: "Maybe using a hash set?"
CodeSage: "Exactly! Hash sets offer O(1) average lookup time. Want to try implementing that approach?"
[**Finals adds:** Real-time complexity detection, natural conversation flow, adaptive follow-up questions]

4 Evaluation Metrics (Weighted Scoring)

Evaluation Dimension	Weight
Conversational Intelligence & Adaptivity	30%
Quality of Analysis & Feedback	25%
Performance & System Reliability	20%
User Experience	15%
Innovation & Creativity	10%

5 Submission Guidelines

To submit your project, please provide the following in a single submission package:

1. **Web Application URL**: Working interface for manual testing
2. **API Documentation**: Postman collection or OpenAPI spec with example calls
3. **Demo Video**: 3-minute recording following the prescribed demo script
4. **Source Code**: GitHub repository link (for finals qualification only)

Build the Future of Technical Assessment

May the best CodeSage win!