

AI-Interviewer Agent – Technical Report

Abstract

This project develops an AI-based Interviewer system that conducts HR, Technical, and Behavioral interviews using LLMs and resume-aware RAG. It evaluates text responses and whiteboard diagrams, adapts question difficulty, and generates explainable scores and personalized feedback for realistic interview assessment.

1. System Architecture

The AI Interviewer is a modular, state-driven interview simulation platform designed to conduct realistic multi-round interviews with adaptive difficulty, personality-based behavior, multimodal inputs, and structured evaluation.

High-Level Components

Frontend (Streamlit UI)

- Collects candidate details and resume
- Displays interview chat interface.
- Provides whiteboard interface for technical rounds
- Shows post-interview analytics, feedback, and learning plan
- Allows downloadable DOCX report generation

Interview Orchestration Layer (LangGraph)

- Manages interview flow using a finite-state graph
- Controls transitions between HR → Technical → Behavioral → Closing
- Enforces hard gates (e.g., technical round failure stops interview)
- Maintains strict single-pass finalization to prevent duplicate summaries

LLM Layer (Groq + Vision Models)

- Text reasoning via Groq LLaMA-3.1 models
- Vision reasoning for whiteboard analysis using OpenRouter / LLaMA vision
- Separate prompts for intent detection, answer judgment, reactions, hints, and final evaluation

Evaluation & Analytics Engine

- Multi-dimension scoring (technical depth, clarity, confidence, communication, correctness)
- Hint penalties and whiteboard bonuses
- Topic-level performance breakdown
- Round-wise and final verdict computation

RAG Layer (ChromaDB)

- Resume is indexed into a vector database
- Resume context is retrieved during question generation
- Ensures interview questions are personalized and grounded

Learning Plan Generator

- Converts interview weaknesses into a personalized roadmap
- Suggests practice tasks and curated learning resources
- Adapts recommendations based on interviewer personality

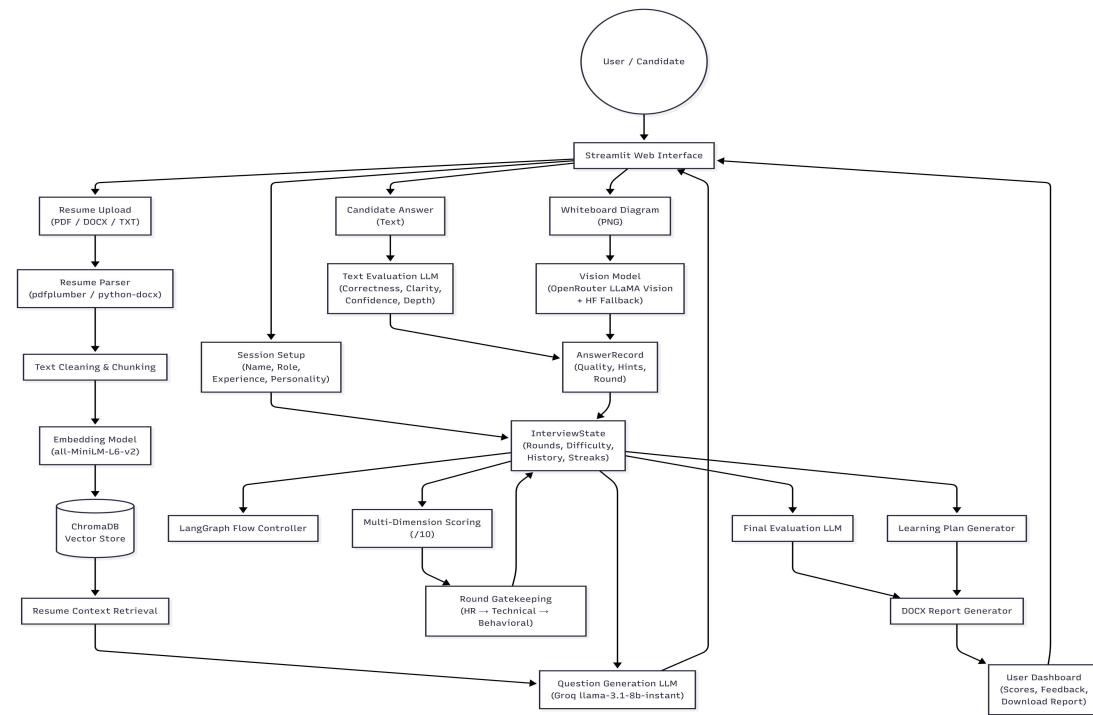


Fig: System architecture

1.1 Candidate Input + Resume

The interview begins when the candidate enters basic details and uploads a resume through the Streamlit interface. The candidate also selects the interviewer personality and role. This information is captured as the initial input and stored for use throughout the interview lifecycle.

1.2 Resume Parsing & Indexing (RAG)

The uploaded resume is parsed to extract text from supported formats such as PDF, DOCX, or TXT. The extracted text is cleaned, chunked, and converted into vector embeddings. These embeddings are stored in a vector database and later retrieved during question generation to ensure that interview questions are personalized and grounded in the candidate's background.

1.3 InterviewState Initialization (LangGraph Control)

After resume indexing, the system initializes a structured InterviewState object. This object stores all runtime information including candidate profile, selected interviewer personality, current round, difficulty level, answer history, hint usage, and scoring metadata. A LangGraph state machine uses this state to control interview flow and enforce strict transition rules.

1.4 HR Round (Personality-Aware Evaluation)

The interview starts with the HR round. Personality-aware HR questions are generated to assess communication skills, clarity of expression, and confidence. Candidate responses are evaluated, and limited retries or follow-up questions allowed for weak answers. Completion of this round triggers an automatic transition to the technical stage.

1.5 Technical Round (Hard Gate + Whiteboard Vision)

The technical round acts as a hard gate for the interview. Resume-aware technical questions are generated using retrieved resume context, interviewer personality, and adaptive difficulty logic. Candidate answers are evaluated for correctness and technical depth. Poor performance beyond a defined threshold immediately terminates the interview.

During this round, candidates may optionally upload whiteboard diagrams. These diagrams are analyzed by a vision-language model, which provides concise structural feedback and generates exactly one follow-up technical question.

1.6 Behavioral Round (Role-Specific Evaluation)

Candidates who pass the technical round proceed to the behavioral round. This stage focuses on role-specific behavioral questions designed to evaluate real-world reasoning, decision-making, and ownership.

1.7 Evaluation & Scoring Engine (Multi-Dimension)

After the interview rounds are completed or terminated early, all recorded answers are aggregated. The system computes scores across multiple dimensions including technical depth, correctness, clarity, confidence, and communication. Hint penalties, repetition checks, and whiteboard bonuses are applied. Round-wise and overall scores are then calculated.

1.8 Final Verdict + Learning Plan

Based on aggregated scores and interview rules, the system determines the final verdict (Selected or Not Selected). In parallel, weak topics are identified and converted into a personalized learning plan consisting of recommended practice tasks and curated learning resources.

1.9 Report Generation & Interview Closure

Finally, a structured interview report is generated containing the interview summary, scores, strengths, improvement areas, final evaluation, and learning plan. The report can be downloaded in DOCX format. The interview is then politely closed, completing the interview process.

2. Interview Flow and Agent Logic

2.1 Interview Rounds

The interview always follows **three fixed rounds**:

1. HR Round

- Personality-aware situational questions
- Controlled retries and follow-ups
- Evaluates communication, clarity, and confidence

2. Technical Round

- Core gatekeeper round
- Adaptive difficulty (levels 1–5)
- Resume-based, system design, coding, and theory questions
- Whiteboard interaction allowed
- Failure immediately ends interview

3. Behavioral Round

- Role-specific behavioral questions
- Enforced minimum and maximum question counts
- Evaluates decision-making, ownership, and reasoning

2.2 State-Driven Control (LangGraph)

The interview is implemented as a LangGraph state machine:

- InterviewState carries all runtime data
- Each user input triggers exactly one graph step
- Hard guards prevent:
 - Duplicate final evaluations
 - Continuing after technical failure
 - Skipping mandatory rounds

2.3 Personality-Based Behavior

The interviewer behavior changes dynamically based on selected personality:

Personality	Behavior
Friendly Coach	Supportive, learning-oriented
Professor	Theory-heavy, formal correctness
Startup Hiring Manager	Practical, real-world trade-offs
Strict FAANG	High difficulty, zero tolerance

Personality affects:

- Question difficulty escalation
- Hint limits
- Strictness of evaluation
- Final verdict logic

2.4 Logical Agents in the AI Interviewer System

The AI Interviewer system is implemented using multiple logical agents. Each agent is responsible for a specific function in the interview lifecycle and operates over a shared InterviewState. The agents are coordinated using a LangGraph state machine to ensure controlled execution, fairness, and explainability.

2.4.1 Interview Orchestrator Agent

- Controls round transitions (HR → Technical → Behavioral)
- Enforces hard technical gate
- Prevents duplicate final evaluations
- Terminates interview when failure conditions are met

2.4.2 Question Generation Agent

- Generates HR, technical, and behavioral questions
- Uses resume context (RAG), role, personality, and difficulty
- Produces follow-up questions for weak answers

2.4.3 Evaluation and Scoring Agent

- Evaluates answers across multiple dimensions
- Applies hint penalties and weak-answer limits
- Computes round-wise and final scores
- Determines final verdict

2.4.4 Intent Detection and Repetition Control Agent

- Detects user intent (answer, hint, unsure, clarification)
- Blocks repeated or reused answers
- Limits excessive hint usage

2.4.5 Whiteboard Vision Agent

- Analyzes uploaded diagrams during technical round
- Provides concise structural feedback
- Generates exactly one follow-up technical question
- Applies capped whiteboard bonus

2.4.6 Learning Plan Generation Agent

- Identifies weak topics after evaluation
- Generates personalized practice tasks
- Suggests curated learning resources

3. Use of Multimodal Inputs (Whiteboard)

3.1 Whiteboard Integration

The system supports diagram-based answers during the Technical round:

- Candidate opens an external drawing canvas
- Exports diagram as PNG
- Uploads the image into the interview

3.2 Vision-Based Analysis

The whiteboard image is analyzed using a vision-language model:

- Identifies visible elements (boxes, arrows, flows)
- Detects missing steps, risks, or design flaws
- Produces concise diagram-specific feedback
- Generates exactly one follow-up technical question

If vision analysis fails, the system safely falls back to text-only reasoning using the candidate's explanation.

4. Memory and State Management Strategy

4.1 InterviewState (Single Source of Truth)

All interview data is stored inside a structured state object:

- Candidate profile and resume
- Current round and question index
- Answer records
- Hint usage and retry counters
- Difficulty level and streak tracking
- Final verdict and evaluation

4.2 Answer Records

Each answer is stored as an `AnswerRecord` containing:

- Question text
- Candidate answer
- Topic and question type
- Quality label (good / weak)

- Hints used
- Round name
- Whiteboard usage metadata

These records drive:

- Final scoring
- Topic breakdown
- Learning plan generation
- Downloadable report content

4.3 Repetition and Concept Memory

The system actively prevents:

- Reusing the same answer across different questions
- Over-helping using repeated hints

It also supports **one-time** concept reminders:

- If a candidate previously explained a concept well
 - The agent encourages applying the same idea to a new scenario
-

5. Design Decisions and Known Limitations

5.1 Key Design Decisions

- **Hard Technical Gate:** Technical failure immediately stops the interview
- **Strict Finalization Guard:** Prevents duplicate summaries or verdicts
- **No Silent Fallbacks:** All LLM failures are explicit and controlled
- **Explainability First:** Every score is traceable to recorded answers
- **Safety-First State Machine:** No undefined transitions or loops

5.2 Known Limitations

- Vision models depend on external APIs and may face latency
- Diagram understanding is limited to visible structural cues
- Learning resources rely on curated mappings when LLM fails
- Interview realism is partially dependent on the reasoning quality and consistency of the underlying language models.
- Offline usage is not supported due to model dependencies

6. Example interview transcripts

 **AI Interviewer (RAG + Whiteboard)**

Name *
yeshu

Role *
Software Engineer

Experience Level *
Entry

Choose Interviewer Personality *
Friendly Coach

Upload Resume * (PDF / DOCX / TXT)
Drag and drop file here
Limit 200MB per file • PDF, DOCX, TXT

 Yesesvini Sai Lanka Resume.pdf 89.4KB 



Fig:6.1 Interview initialization screen used to collect candidate details, configure interviewer personality, and ingest resume for RAG-based question generation.

AI Interviewer

 Interviewer: Friendly Coach

 Good evening, yeshu 

I'm Alex, your interview coach.

I'll interview you in a **Friendly Coach** style.

Let's start.

  HR Round

 Q1 – HR (theory):

Tell me about yourself.

Fig:6.2 Interview start screen showing a time-based personalized greeting, interviewer introduction, selected interviewer personality, and the beginning of the HR round with the first interview question.

 Q4 – HR (theory):

Imagine you've just finished a project and the team lead informs you that the project's deadline has been pushed back by a week. However, the team's morale is at an all-time high due to your team's recent success. How would you handle this news and ensure the team's morale remains high?

 I would first acknowledge and congratulate my colleague for their discovery, emphasizing that their work is valuable and independent. Then, I would clearly communicate my own contributions, highlighting the aspects of my research that are original or distinct. If appropriate, I would propose collaboration or a joint publication to combine insights while giving proper credit, ensuring both our contributions are recognized fairly.

  This looks like the same answer you used for a previous question. Please answer this question with a NEW example or NEW reasoning.

 I would first communicate the news transparently, explaining the reason for the deadline change and emphasizing that it gives us extra time to polish the project. I'd also celebrate the team's recent success to maintain momentum, highlight the progress we've already made, and encourage them to use the extra time to improve quality or explore innovative ideas. This approach keeps morale high while focusing the team on making the most of the opportunity.

 That shows good insight into effective team management.

Fig:6.3 Screen showing detection of a repeated response, a request for a revised answer, evaluation of the improved response, and positive feedback.

  HR Round completed (Score: 8.3/10). Moving to Technical Round.

  Technical Round

 Q1 – Technical (theory):

Write a JavaScript function to find the maximum value in an array of numbers. Use a for loop to iterate through the array, and store the maximum value in a variable. If the array is empty, return null.

 Q1 – Technical (theory):

Can you explain the concept of object-oriented programming (OOP) and its importance in software development, using an example from your experience or a real-world scenario?



Fig:6.4 Screen illustrating the completion of the HR round with score feedback, the automatic transition to the technical round, and the presentation of the first technical interview question.

Fig:6.5 Technical question screen displaying an “Open Whiteboard” button, allowing candidates to visually explain concepts using diagrams.

Q3 — system (theory):

You designed and implemented a Placement Portal using HTML, CSS, and JavaScript for the frontend, and integrated it with a backend powered by Java for seamless college placement processes. Describe the system design, its trade-offs, and any failures you encountered during development.

Whiteboard Feedback

The diagram provided by the candidate lacks a clear overview or high-level system architecture. It only details the frontend and backend components, but does not illustrate how these components interact with each other or with external systems, such as databases or APIs. This omission makes it difficult to understand the overall system design and its trade-offs.

What trade-offs did you make in choosing to use Java for the backend of the Placement Portal, and how do these decisions impact the overall performance and scalability of the system?

I chose Java for the backend because it offers strong performance, stability, and good support for building scalable REST APIs. The trade-off was slower initial development and higher memory usage compared to lighter frameworks. However, this decision improved code structure, concurrency handling, and long-term scalability, making the system more reliable during high user load.

Fig:6.6 Technical round screen illustrating a system design question, automated whiteboard feedback, vision-based follow-up questioning, and candidate response evaluation.

Q4 — system (theory):

You have been tasked with designing a system to manage a college's placement process. Describe how you would design and implement a system to efficiently manage student profiles, resumes, and facilitate seamless communication between students and recruiters. Consider the design trade-offs and potential failures that could arise from your approach.

can i get hint

Hint: Here's a hint to get you started:

Consider designing a microservices-based system with separate components for student profile management, resume storage, and communication channels, and think about how you can leverage APIs for seamless integration between these components.

i am not sure

Okay, let's move on.

Fig:6.7 Technical round screen showing how the system gives a hint when the candidate asks for help, detects when the candidate is unsure, and automatically moves to the next question.

Q7 — databases (coding):

You have a database with a table 'students' having columns 'id', 'name', 'age', and 'department'. Write a SQL query to retrieve the names and ages of students from the 'department' that has the maximum number of students.

SELECT planet_name, galaxy FROM stars WHERE brightness > 1000;

This answer is not relevant to the task at hand.

Wrong answer. Hint: It looks like your query is a bit off track. Can you think of the first step to get the department with the maximum number of students, before trying to retrieve their names and ages?
Try once more.

SELECT name, age FROM students WHERE department = (SELECT department FROM students GROUP BY department ORDER BY COUNT(*) DESC LIMIT 1);

That shows clear understanding.

Fig:6.8 Technical coding question screen illustrating incorrect answer detection, guided hint feedback, a second attempt, and confirmation of the correct SQL solution.

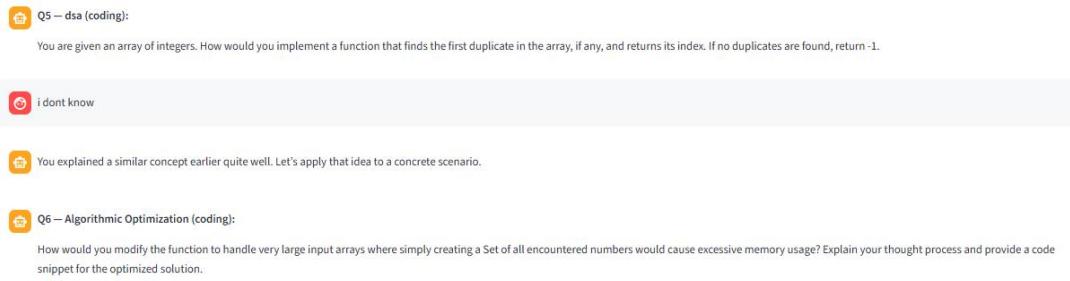


Fig:6.9 Screen showing that the system detects when the user is unsure, gives a one-time reminder of a previously explained idea, and then moves to a follow-up question about improving the solution.

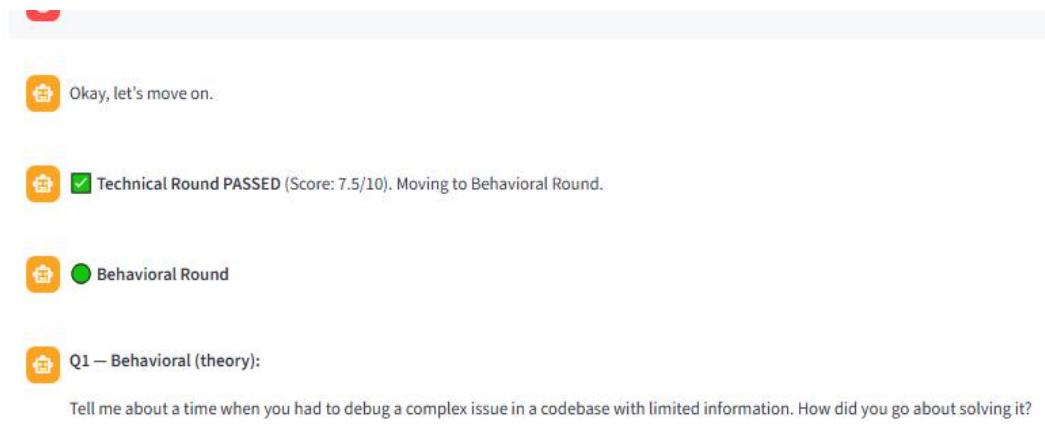


Fig:6.10 Screen showing technical assessment completion with score feedback, automatic transition to the next evaluation stage, and presentation of the first behavioral question.

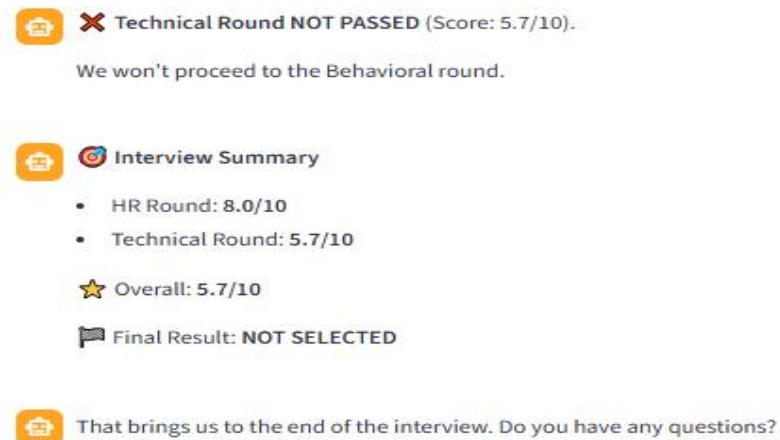


Fig:6.11 Screen showing interview termination after technical evaluation, score summary, and final non-selection outcome.

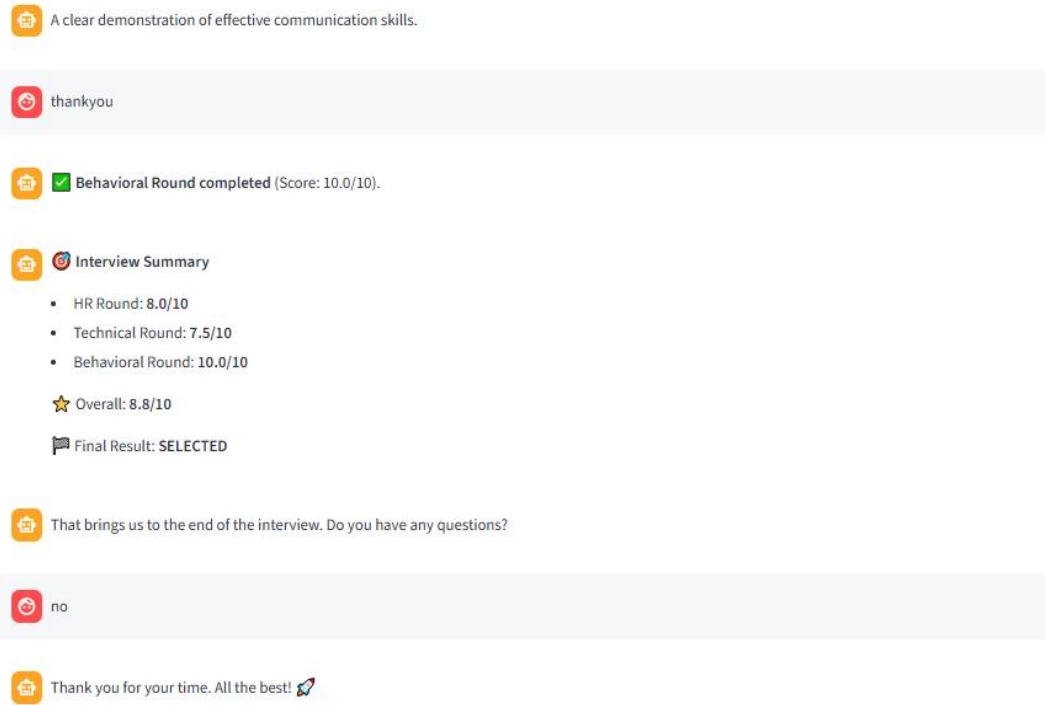


Fig:6.12 Final interview screen showing round-wise score summary, overall evaluation, selection decision, and polite interview closure.

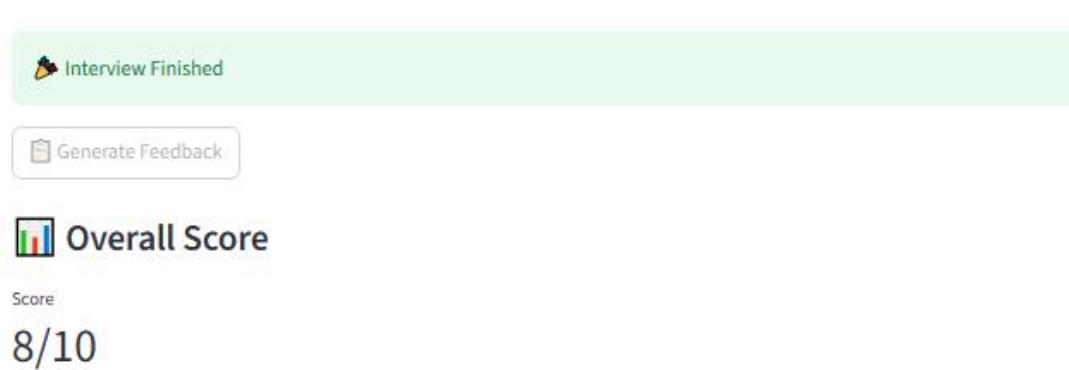


Fig:6.13 Screen showing interview completion status and the overall score of the candidate.

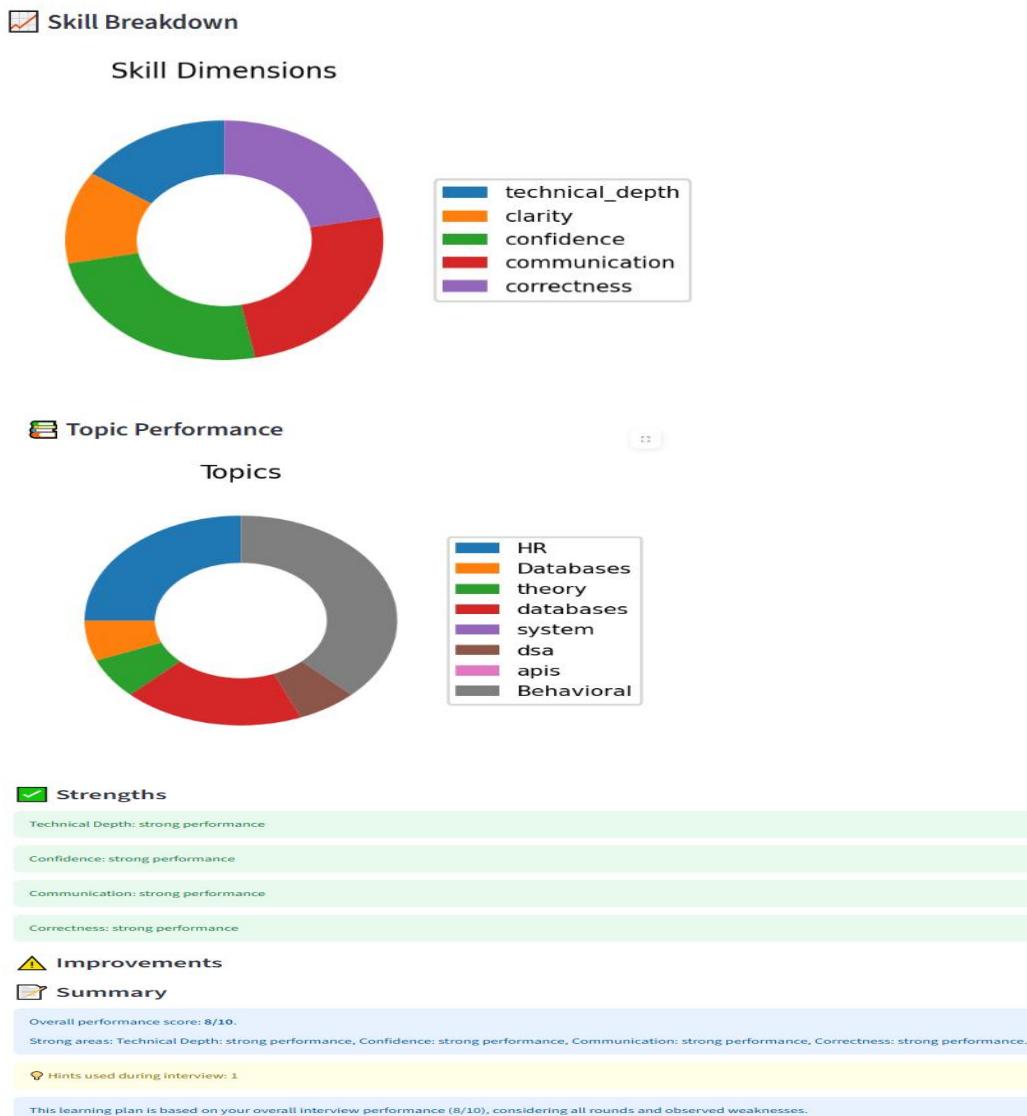


Fig:6.14 Post-interview evaluation dashboard showing skill breakdown, topic performance, strengths, summary, personalized learning roadmap, and recommended resources.

 Download Interview Report (DOCX)

Fig:6.15 Screen showing the option to download the interview report in DOCX format.

Conclusion

This project presents an AI-based interview system that conducts structured HR, Technical, and Behavioral interviews using resume-based personalization and adaptive evaluation. The system provides transparent scoring, identifies strengths and weaknesses, and generates a personalized learning roadmap with recommended resources. By combining agent-driven orchestration, multimodal whiteboard analysis, and explainable feedback, the platform delivers a realistic and effective interview assessment and preparation experience.