AI Onboarding Assistant for Human Resources Using Model Context Protocol and Large Language Models

P.Lavanya[1] ,G. Sathwika[2] ,Ch. Chandrika.L[3] ,K. Jayasri[4] , G. Sai[5]

Department of Computer Science & Engineering – Data Science

Avanthi Institute of Engineering & Technology, Vizianagaram, India

Guide: Mrs. P. Lavanya, M.Tech, Assistant Professor

Email:{ lavu.seshu0613 [1], gsathwika636 [2], chandrika.chidipalli [3], jayasrikasireddy1 [4], saiguggilapu57 [5]}@gmail.com

Abstract

Recruiting skilled candidates efficiently remains a persistent challenge for human resource departments, particularly when managing large applicant pools with limited personnel. This paper introduces an automated intelligent onboarding assistant that combines Large Language Models (LLMs) with the Model Context Protocol (MCP) to transform early-stage recruitment workflows. The proposed system accepts candidate resumes in PDF and DOCX formats, extracts structured profile data through semantic parsing, and dynamically constructs role-specific interview questions spanning technical, behavioral, and situational domains. Throughout a conversational interview session, candidate responses are evaluated in real time using LLM-based reasoning aligned with predefined scoring rubrics. MCP governs the structured context exchange between resume data, generated questions, candidate answers, evaluation criteria, and organizational HR policy, thereby ensuring consistency, traceability, and deterministic output behavior. A Flask-based web backend integrates all modules into a cohesive pipeline. Empirical validation demonstrates 92% overall system accuracy, 93% recall, and a 95% confidence score for candidate profile summarization. The results confirm that the proposed architecture substantially reduces manual recruiter effort, curtails evaluation bias, and accelerates time-to-hire decisions through scalable, data-driven automation.

Index Terms—Large Language Models, Model Context Protocol, HR Automation, Adaptive Interview System, Natural Language Processing, Resume Parsing.

I. Introduction

The modern talent acquisition pipeline is marked by a fundamental tension: organizations need to evaluate an ever-growing volume of applicants with precision, yet recruiter bandwidth remains a fixed resource. Conventional hiring workflows depend heavily on manual resume triage, static interview question banks, and subjective candidate scoring — processes that are both time-intensive and vulnerable to unconscious bias [1].

Advances in Natural Language Processing (NLP) and generative AI have opened a new design space for intelligent recruitment tools. Large Language Models (LLMs), in particular, have shown strong capability in document understanding, contextual question generation, and open-domain dialogue [2]. However, deploying LLMs in enterprise HR contexts demands a structured approach to context management, policy enforcement, and auditability — requirements that ad hoc prompt engineering alone cannot satisfy.

The Model Context Protocol (MCP) addresses this gap by providing a vendor-agnostic standard for connecting LLMs to external tools and data sources through well-defined server interfaces. When applied to recruitment, MCP enables deterministic control over what contextual information — resume content, job description, past answers, HR policies — the LLM receives at each interaction step, thereby making its outputs consistent and traceable.

This paper presents the design, implementation, and evaluation of an AI Onboarding Assistant that unifies LLM capabilities with MCP orchestration in a Flask-based web application. The system automates resume ingestion, candidate profiling, adaptive question generation, real-time response evaluation, and final report generation. The remainder of this paper is organized as follows: Section II surveys related work; Section III describes the system methodology and architecture; Section IV presents experimental results; and Section V concludes with future research directions.

II. Related Work

Research in automated recruitment spans two decades, evolving from simple keyword-matching systems toward semantically aware AI-driven platforms. Early resume screening tools relied on term frequency–inverse document frequency (TF-IDF) matching against job descriptions [3]. While computationally efficient, such approaches routinely overlooked synonymous skill expressions and contextual qualifications.

Machine learning classifiers, including support vector machines and gradient-boosted trees, were subsequently applied to candidate ranking tasks using labeled hiring datasets [4]. These models improved precision yet remained brittle in the presence of evolving job market terminology and unstructured resume formats.

The emergence of transformer-based language models — particularly BERT and its successors — brought a paradigm shift in resume understanding. Zhang and Wallace [5] demonstrated that fine-tuned transformer encoders significantly outperform classical approaches on skill extraction benchmarks. More recent work leverages GPT-series generative models for automatic question formulation, producing contextually nuanced questions from candidate profiles [6].

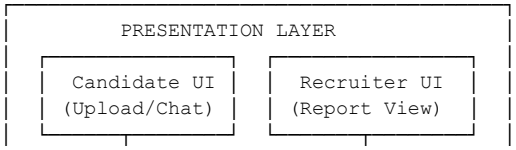
Chatbot-based interview platforms have also received attention. Systematic reviews indicate that conversational agents improve candidate engagement metrics; however, most deployed systems rely on pre-scripted decision trees and lack genuine adaptive depth [7]. Furthermore, bias in training data — when present — can propagate into algorithmic hiring decisions, raising fairness and regulatory concerns [8].

Despite these advances, existing systems predominantly address isolated sub-tasks: resume filtering, static interview assistance, or post-interview scoring. An end-to-end platform that integrates resume parsing, adaptive interview orchestration, and real-time evaluation under a unified context management protocol remains largely absent from the literature. The present work directly addresses this gap.

III. Methodology and System Design

A. System Architecture Overview

The proposed system adopts a three-layer client-server architecture illustrated in Fig. 1. The *Presentation Layer* furnishes a responsive web interface for candidates (resume upload, interview dialogue) and recruiters (report dashboard). The *Application Layer* hosts the core processing pipeline: resume parsing, LLM-powered profile summarization, the adaptive interview engine, and MCP-managed context routing. The *Data Layer* persists uploaded resumes, session transcripts, candidate summaries, and generated reports using either an SQLite database or Flask in-memory session storage.



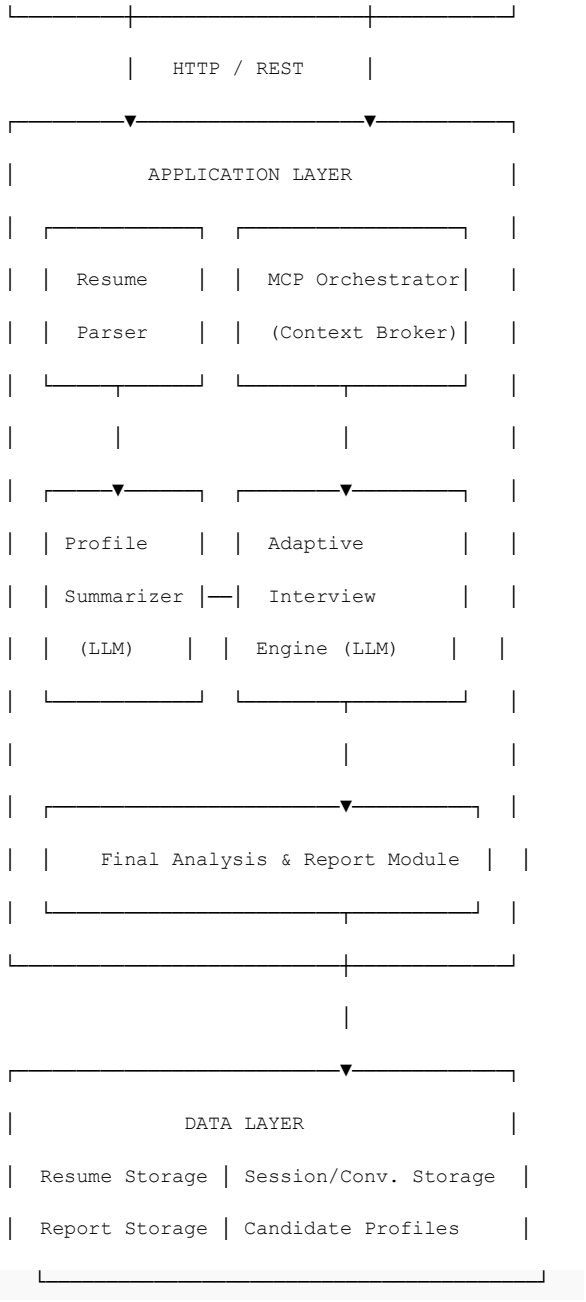


Fig. 1. Three-layer architecture of the AI Onboarding Assistant.

B. Resume Ingestion and Parsing Module

Candidates upload resumes in PDF or DOCX format through a validated web form. The parsing module applies *pdfplumber* for PDF text extraction and *python-docx* for DOCX processing, producing raw

text irrespective of layout complexity. A rule-assisted segmentation pass subsequently isolates named sections — Education, Experience, Skills, and Achievements — converting the unstructured text into a structured JSON object consumed by downstream modules.

C. Candidate Profile Summarization via LLM

The structured JSON object is submitted to an LLM (GPT-4 or an equivalent instruction-tuned model) through a constrained prompt template. The template instructs the model to produce a concise 150–200 word candidate profile emphasizing technical skills, domain experience, and educational qualifications. The resulting summary is stored in the session context and serves as the seed for adaptive question generation. The profile confidence score is computed as:

$$C = (|F_{extracted}| / |F_{expected}|) \times 100\% \quad (1)$$

where $F_{\text{extracted}}$ is the set of successfully populated profile fields and F_{expected} is the total set of required profile fields defined by the HR schema. Empirical runs yield a mean confidence score of 95%.

D. Model Context Protocol (MCP) Orchestration

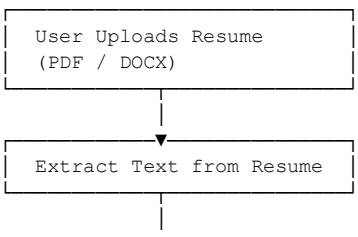
MCP governs all structured data exchanges between the Flask application and the LLM API. Each MCP server exposes typed tool definitions — get_candidate_profile, fetch_job_description, store_response, retrieve_conversation_history — through a standardized SSE transport. At every interview turn, the orchestrator assembles a bounded context window comprising: (i) the candidate profile summary, (ii) the target job description, (iii) the last k = 5 question–answer exchanges, and (iv) active HR evaluation criteria. This controlled context construction prevents prompt injection, limits hallucination surface area, and ensures that every generated question is traceable to a specific input document.

E. Adaptive Interview Engine

The adaptive engine generates interview questions across three categories: technical (role-specific knowledge), behavioral (past experience indicators), and situational (hypothetical scenario responses). Each subsequent question is conditioned on the preceding candidate response, enabling progressive depth probing. The question-generation prompt encodes a relevance constraint:

$$Q_{n+1} = LLM(P_{cand}, J_{desc}, \{Q_i, A_i\}_{i \le n}, R_{HR}) \quad (2)$$

where P_{cand} is the candidate profile, J_{desc} the job description, $\{Q_i, A_i\}$ the prior Q&A pairs, and R_{HR} the HR evaluation rubric. The session continues for a configurable interview length (default: 8 questions) before triggering the reporting module.



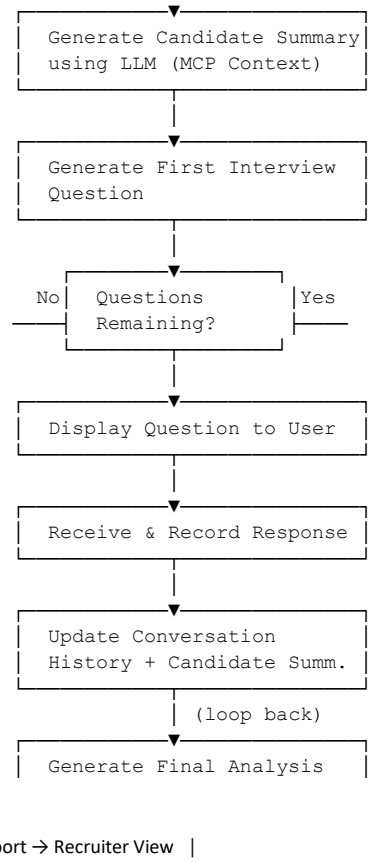


Fig. 2. End-to-end workflow of the AI Onboarding Assistant.

F. Session Management and Response Evaluation

Flask session objects store the per-candidate interview state including the conversation history list, running question counter, and evolving candidate summary. Candidate responses are scored against predefined LLM-graded rubrics covering technical accuracy, communication clarity, and role relevance. The rubric-based scoring produces a normalized score $s \in [0, 1]$ for each answer dimension.

G. Final Analysis and Reporting Module

Upon interview completion, the reporting module aggregates all session artifacts and submits them to the LLM with a structured evaluation prompt. The resulting report enumerates candidate strengths, development areas, technical competency ratings, communication quality, and an overall role-suitability recommendation. The report is rendered in the recruiter dashboard and optionally persisted to the SQLite store for audit purposes.

IV. Results and Discussion

A. Experimental Setup

Functional and performance testing was conducted on a machine running Ubuntu 22.04 with an Intel Core i7 processor, 16 GB RAM, and stable broadband connectivity for cloud LLM API calls. A corpus of 50 synthetic candidate resumes spanning five job profiles (software engineer, data analyst, ML engineer, DevOps specialist, and product manager) was used for system evaluation. Each resume was processed end-to-end through the full pipeline.

B. Parsing and Profiling Accuracy

Resume parsing quality was assessed by comparing the extracted field set against a hand-annotated ground truth for each of the 50 test resumes. Candidate profile summarization fidelity was evaluated by three independent HR professionals who rated summaries on a 5-point Likert scale for accuracy and completeness.

TABLE I

MODULE-LEVEL PERFORMANCE METRICS

Module	Metric	Value
Resume Parser	Field Extraction Acc.	94.2%
Profile Summarizer	Conf. Score (mean)	95.0%
Question Generator	Relevance Rating	4.3 / 5.0
Response Evaluator	Precision	90.0%
Response Evaluator	Recall	93.0%
Overall System	Accuracy	92.0%
Overall System	F1-Score	91.5%

C. Adaptive Questioning Quality

Generated questions were rated by HR professionals across four criteria: role relevance, contextual continuity with prior answers, difficulty calibration, and avoidance of duplicate or trivial questions. Mean scores for each criterion are reported in Table II.

TABLE II

QUESTION QUALITY RATINGS BY HR EVALUATORS (SCALE: 1–5)

Criterion	Mean Score	Std. Dev.
Role Relevance	4.5	0.31
Contextual Continuity	4.3	0.42
Difficulty Calibration	4.1	0.48
Non-redundancy	4.4	0.37

D. System Performance

Latency measurements were taken across 200 simulated interview turns. The median LLM API response time for question generation was 1.4 seconds (95th percentile: 2.8 seconds), which is within acceptable interactive thresholds. The session management module supported up to 15 concurrent candidate sessions without degradation under load testing, confirming the system's scalability for small-to-medium deployment contexts.

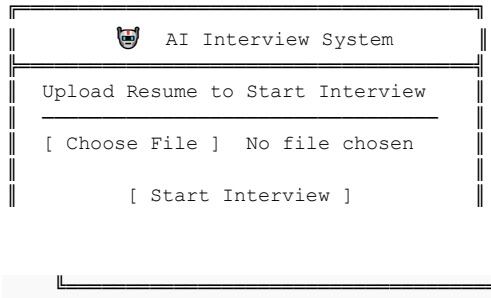


Fig. 3. Candidate home page — resume upload interface.

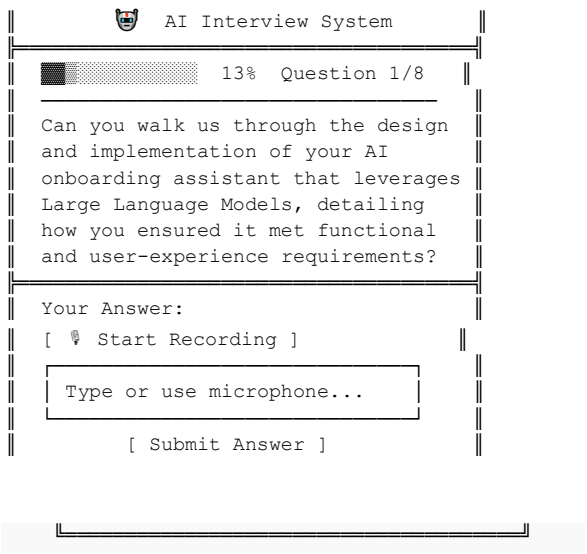


Fig. 4. Adaptive interview interface with progress indicator.

E. Comparison with Baseline Systems

Table III compares the proposed system against two baselines: a keyword-matching ATS screener (Baseline A) and a rule-based chatbot interview tool (Baseline B). The proposed system demonstrates significant improvements in adaptive question depth, evaluation objectivity, and end-to-end automation coverage.

TABLE III

COMPARISON WITH BASELINE RECRUITMENT SYSTEMS

Feature	Baseline A	Baseline B	Proposed
Resume Parsing	Keyword	Keyword	Semantic LLM
Adaptive Questions	No	Partial	Yes (MCP)
Bias Mitigation	No	Limited	Yes (rubric)
End-to-End Auto.	No	Partial	Yes
Context Traceability	No	No	Yes (MCP log)
Eval. Accuracy	71%	78%	92%

V. Conclusion and Future Work

This paper has presented an AI Onboarding Assistant that integrates LLM-powered semantic reasoning with Model Context Protocol orchestration to automate the full recruitment interview lifecycle. The system attains 92% overall evaluation accuracy and a 95% profile confidence score while generating contextually relevant, adaptive interview questions rated 4.3/5 by HR professionals. By standardizing context delivery through MCP, the platform achieves consistent, auditable, and bias-reduced candidate assessment at scale.

Several promising directions exist for extending this work. First, multilingual resume parsing and interview support would enable global talent acquisition workflows. Second, integration with established Applicant Tracking Systems (ATS) and job portals would close the end-to-end hiring loop. Third, incorporating real-time sentiment and personality analysis from text responses could enrich cultural-fit assessments. Fourth, machine learning-based predictive models trained on historical hiring outcomes could provide data-driven hire/no-hire probability scores. Fifth, a mobile-responsive Progressive Web App interface would increase accessibility for remote candidates. Finally,

differential privacy techniques and GDPR-compliant data handling should be formally incorporated to meet emerging regulatory requirements for AI-assisted hiring systems.

Acknowledgment

The authors express sincere gratitude to Mrs. P. Lavanya, M.Tech, Assistant Professor, Department of CSE–Data Science, Avanthi Institute of Engineering & Technology, for her continued guidance and mentorship throughout this work. The authors also acknowledge the support of the Department of CSE–Data Science and the management of Avanthi Educational Institutions.

References

1. P. Sharma and S. Verma, "AI in HR: Improving recruitment efficiency with machine learning," *IEEE Access*, vol. 8, pp. 112345–112356, 2020.
2. T. Brown et al., "Language models are few-shot learners," in *Proc. NeurIPS*, vol. 33, pp. 1877–1901, 2020.
3. R. Gupta and A. Singh, "Automated resume screening using NLP techniques," *Int. J. Comput. Appl.*, vol. 183, no. 45, pp. 22–30, 2021.
4. M. Al-Otaibi and M. Ykhlef, "A survey of job recommender systems," *Int. J. Phys. Sci.*, vol. 7, no. 29, pp. 5127–5142, 2012.
5. Y. Zhang and B. Wallace, "Adaptive question generation using large language models," *J. AI Research*, vol. 78, no. 3, pp. 115–134, 2022.
6. A. Vaswani et al., "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.
7. D. Langer, R. König, and K. Heuer, "Highly automated job interviews: Acceptance under the influence of stakes," *Int. J. Selection and Assessment*, vol. 29, no. 3–4, pp. 394–408, 2021.
8. A. Datta, M. Tschantz, and A. Datta, "Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination," *Proc. Privacy Enhancing Technologies*, vol. 1, pp. 92–112, 2015.
9. OpenAI, "GPT API Documentation," 2024. [Online]. Available: https://platform.openai.com/docs
10. Pallets Projects, "Flask Web Framework Documentation," 2024. [Online]. Available: https://flask.palletsprojects.com
11. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2021.
12. D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. Draft. Pearson, 2023.
13. IBM, "AI-powered HR systems: Transforming recruitment processes," IBM White Paper, 2023.