JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

IRIS: Interview Relevance and Interactive Simulation

Sheela Verma[1]*, Prikshit Singh[2], Priyanshu Singh[3], Raunit Kumar[4]

1*Assistant Professor: Bhilai Institute of Technology, Raipur
2,3,4 B. Tech Scholar: Bhilai Institute of Technology, Raipur

Abstract: This paper presents IRIS (Interview Relevance and Interactive Simulation), a novel AI-powered system designed to automate and enhance document-based interviews. IRIS provides an end-to-end solution integrating contextual Retrieval-Augmented Generation (RAG), automated response evaluation via the HARP (Human Answer Relevance Protocol), computer vision-based behavior monitoring, and Text-to-Speech (TTS) capabilities. The system utilizes a modular architecture orchestrated by an LLM agent (Interviewer Host) interacting with specialized backend services through Model Context Protocol (MCP) servers. Key components include a contextual RAG system employing hybrid retrieval (semantic FAISS/Qwen embeddings and lexical BM25) and neural reranking for relevant question generation from source documents; the HARP service using LLMs for objective, multi-parameter evaluation of candidate answers against questions and expected responses; an attention detection service leveraging YOLOv8, Mediapipe, and L2CS-Net for behavioral insights; and an AllTalk/XTTS-based TTS module accessed via MCP for vocalizing interview content. Experimental results demonstrate that IRIS significantly improves interview consistency, evaluation objectivity, and time efficiency compared to traditional methods. The system's architecture supports scalability and adaptability across various interview scenarios and document types, offering a comprehensive and interactive platform for standardized and insightful document-centric assessments.

Keywords **- Natural Language Processing, Model Context Protocol, Retrieval-Augmented Generation, Computer Vision, Automated Evaluation, Interview Systems, Text-to-Speech.**

I. INTRODUCTION

Interviews centered around specific document content are fundamental in various domains, including educational assessments, technical skill evaluations, compliance checks, and research contexts. However, traditional approaches to such interviews often suffer from significant limitations. These include inconsistency in the relevance and depth of questions asked, subjectivity and variability in human evaluation of free-text responses, difficulties ensuring the interview remains focused on the provided document, and a lack of objective methods to gauge candidate engagement or attention during the process. Consequently, traditional methods can be time-consuming, costly, and may not provide a consistently fair or accurate assessment of a candidate's understanding of the document content.

Recent advancements in Large Language Models (LLMs), Retrieval-Augmented Generation (RAG) [1], [3], [4], and Model Context Protocol (MCP) [12] offer powerful new tools for developing intelligent systems capable of understanding complex documents and interacting dynamically. RAG techniques allow LLMs to ground their responses in retrieved information, improving factual accuracy [3], while MCP enables LLMs to interact with external tools and APIs, extending their capabilities beyond text generation [12]. Simultaneously, progress in computer vision, particularly with real-time object detection frameworks like YOLO [2], [9] and pose/gaze estimation techniques, allows for automated monitoring and analysis of human behavior [8].

This paper introduces IRIS (Interview Relevance and Interactive Simulation), an integrated system that leverages these technological advancements to create a comprehensive, automated solution for document-based interviews. IRIS aims to overcome the limitations of traditional methods by combining:

1. **Contextual RAG:** An enhanced RAG system processing source documents (PDFs), preserving document context during chunking, and using a hybrid retrieval strategy (FAISS/Qwen semantic search + BM25 lexical search) with neural reranking to generate highly relevant, grounded interview questions and expected answers.
2. **Automated Evaluation (HARP):** The Human Answer Relevance Protocol, implemented as a dedicated service using LLMs, providing objective, multi-dimensional evaluation of candidate answers against the question and RAG-generated responses based on criteria like factual accuracy, relevance, and completeness.
3. **Behavior Monitoring:** A computer vision module utilizing YOLOv8, Mediapipe, and L2CS-Net analysing the candidate's webcam feed to assess attention levels (gaze tracking, drowsiness detection), providing metrics contributing to the overall assessment.
4. **MCP-Driven Orchestration**: An LLM-based Interviewer Host managing the interview flow by interacting with dedicated backend services via specialized MCP servers according to predefined instructions, enabling complex workflow execution.

5. **Text-to-Speech (TTS) Integration:** An AllTalk/XTTS engine accessed via MCP allowing the Interviewer Host to vocalize questions and feedback, enhancing the naturalness and accessibility of the simulation.

The primary contributions of this work lie in the novel integration of these diverse AI technologies within a unified, MCP-orchestrated framework specifically tailored for document-based interviewing, the development of the Contextual RAG technique enhancing retrieval relevance for this task, and the empirical validation of the system's effectiveness compared to traditional approaches.

IRIS provides a standardized yet flexible interview experience, automating question generation, response capture, objective evaluation, behavioral assessment, and interactive vocalization. This paper details the system's architecture, core technologies, and presents experimental results demonstrating significant improvements in interview consistency, evaluation objectivity, and efficiency. The modular design ensures adaptability for diverse interview scenarios and document types, potentially transforming how knowledge and comprehension are assessed in educational, professional, and research settings.

II. Related Work

A. Retrieval-Augmented Generation Systems

Retrieval-Augmented Generation (RAG) combines information retrieval with text generation to enhance the factual accuracy and relevance of LLM outputs. Lewis et al. [3] introduced the RAG framework, demonstrating improved performance on knowledge-intensive tasks. Subsequent work by Guu et al. [4] expanded on this approach with REALM, which uses unsupervised pre-training to learn better retrievers. Recent advances by Khattab et al. [5] introduced ColBERT, a neural ranking architecture that provides efficient and contextualized ranking of documents. While these approaches demonstrate the effectiveness of RAG for knowledge retrieval, they primarily focus on factual question answering rather than document-specific interviewing. Our work extends RAG to the interview domain by incorporating document context and metadata to generate more relevant questions.

B. Automated Response Evaluation

Automated evaluation of free-text responses has been explored in educational technology and natural language processing. Kumar et al. [6] developed automated grading systems for short-answer questions using rubric-based approaches. More recently, Zhao et al. [7] leveraged pre-trained language models for more nuanced evaluation of responses. The emergence of large language models has further advanced automated evaluation capabilities. Wang et al. [8] demonstrated that LLMs can evaluate responses with near-human accuracy when provided with appropriate rubrics and examples. Our work builds on these advances by implementing a custom evaluation framework (HARP) that combines multiple assessment metrics tailored to interview contexts.

B. Computer Vision for Behavior Monitoring

Computer vision techniques have been applied to monitor user behavior in various contexts, including education and assessment. Pei et al. [9] developed systems for tracking gaze and attention during computer-based testing. The introduction of real-time object detection frameworks like YOLO has enabled more efficient behavior monitoring. Redmon et al. [10] introduced YOLO, which has evolved through multiple versions to the current state-of-the-art YOLO v8 [2]. Our work integrates YOLO v8 for monitoring candidate behavior during interviews, focusing on attention tracking and detecting potential cheating behaviors. This integration provides a more comprehensive assessment of candidate performance beyond verbal responses.

III. System Architecture

A. Overview

The IRIS system employs a modular, service-oriented architecture designed to separate concerns and facilitate scalability. It integrates several specialized components orchestrated primarily through Model Context Protocol (MCP) servers, enabling an AI host to conduct complex, interactive, and evaluated interviews based on document content. The architecture ensures seamless interaction between natural language processing, information retrieval, automated evaluation, text-to-speech, and computer vision modules. Figure 1 illustrates the high-level architecture and data flow between the core components.

The main components of the IRIS architecture are:

1. **Interviewer Host (e.g., Claude):** An LLM acting as the conversational interface for the candidate. It drives the interview flow based on predefined system prompts. Crucially, it does not generate questions, evaluations, or speech itself but utilizes tools provided by MCP servers to interact with backend services. It is responsible for presenting questions (textually and auditorily via TTS), receiving answers, and displaying final results, including visualizations.
2. **Interview Assistant MCP Server:** This server acts as a dedicated tool provider for managing the interview lifecycle. It exposes MCP tools (`start_interview`, `submit_answer`, `get_interview_results`, `upload_document`) that the Interviewer Host calls to manage sessions and document handling via the Interview & RAG Service backend. This server translates the LLM's requests into specific API calls to the Interview & RAG Service backend. It also maintains a local cache for session information as a fallback.
3. **HARP MCP Server (HARP-Evaluator):** This server provides MCP tools (`evaluate_answers`, `evaluate_answers_batch`, `save_evaluation_results`) focused specifically on evaluating candidate responses. The Interviewer Host invokes these tools after the interview concludes, passing the collected question-answer pairs to the HARP Evaluation Service backend via its API.
4. **AllTalk TTS MCP Server**: This server exposes MCP tools (`generate_speech`, `stream_speech`, `list_voices`) enabling the Interviewer Host to synthesize speech from text. It bridges the Host's requests to the AllTalk TTS backend API, facilitating auditory output for questions and feedback.
5. **Interview & RAG Service (FastAPI Backend):** This is the core backend engine providing a REST API for several functions:

- **Document Management:** Handling uploads (`/upload-document`) and storing documents locally.

- **Embedding & RAG:** Processing documents using an `EmbeddingManager` to create and cache Qwen embeddings in a FAISS vector store. It initializes and manages the `ContextualHybridRetriever`-based RAG chain for each session.
- **Question Generation:** Leveraging the session's RAG chain to generate contextually relevant questions and expected answers based on the processed document.
- **Session Management:** Starting interviews (`/interview/start/{session_id}`), managing state (current question, answers), and storing raw interview results locally as JSON files.

6. **HARP Evaluation Service (FastAPI Backend):** A dedicated microservice responsible for the objective evaluation of candidate answers. It receives requests via its API (`/evaluate/`) from the HARP MCP Server, compares the candidate's answer to the question and RAG-generated answer using a `LlamaSimilarityEvaluator` (LLM-based), assesses based on configurable parameters (e.g., accuracy, relevance) and scoring formats, and returns detailed JSON results.
7. **Attention Detection Service (FastAPI Backend):** An independent computer vision service responsible for monitoring candidate behavior during the interview via webcam. It uses YOLOv8 for object/interaction detection, Mediapipe for facial landmarks, and L2CS-Net for gaze tracking to assess attention levels (e.g., gaze direction, drowsiness detection). It exposes API endpoints (/start, /stop) to control monitoring sessions and provides metrics (e.g., attention percentage, distraction time) that, as specified in the system design (IRIS System Overview Document), influence the candidate's overall result
8. **AllTalk TTS Backend API:** An external or local service running the AllTalk/XTTS engine, performing the actual speech synthesis based on requests from the AllTalk TTS MCP Server. This component is a crucial external dependency for the IRIS system's speech synthesis capability. It represents a dedicated web service, running either locally on the same infrastructure as IRIS or potentially as a remote service, whose sole responsibility is to perform the computationally intensive task of converting text into audible speech.

- **Core Engine (AllTalk/XTTS):** The service is built upon the AllTalk project's implementation, which leverages the powerful XTTS (specifically XTTSv2) engine developed by Coqui.ai. XTTS is known for generating high-quality, natural-sounding speech across multiple languages. A key feature of XTTS is its strong voice cloning capability, allowing it to mimic specific voices from short audio samples, although the primary use in IRIS seems focused on using pre-defined available voices.
- **Functionality via API:** The backend exposes a RESTful API, primarily designed to be consumed by the AllTalk TTS MCP Server. Key functionalities exposed through this API include:
- **Speech Generation (/api/tts-generate):** Takes text input along with parameters like the desired voice (character_voice_gen), language, and potentially settings for narration or text filtering. It synthesizes the audio asynchronously, saves it as a file (e.g., WAV), and returns a JSON response containing metadata, including a status (generate-success or generate-failure) and often a URL pointing to the generated audio file accessible via the service's web server.
- **Speech Streaming (/api/tts-generate-streaming):** Designed for lower-latency applications, this endpoint takes text and parameters and streams the generated audio bytes back directly in the HTTP response body. This allows the client (the MCP server, which would then relay it) to play the audio as it's being received, suitable for real-time conversational interaction.
- **Voice Management & Status:** Provides utility endpoints like /api/voices to list available voice models/files installed on the backend, /api/previewvoice to generate a sample from a specific voice, and /api/ready as a health check to confirm the service and the underlying TTS model are loaded and operational.

9. Supporting Components:

- **Document Storage:** Local directory (`./documents`) for uploaded files.
- **Vector Store:** FAISS index files for efficient semantic search.
- **Results Storage:** Local JSON files (`./results`) for raw interview data and potentially evaluation/behavioral metric outputs.
- **Supabase:** Potentially used for persistent storage of session metadata, final results, or user data, enhancing data management beyond local files.

Figure 1 illustrates the high-level architecture of the IRIS system.

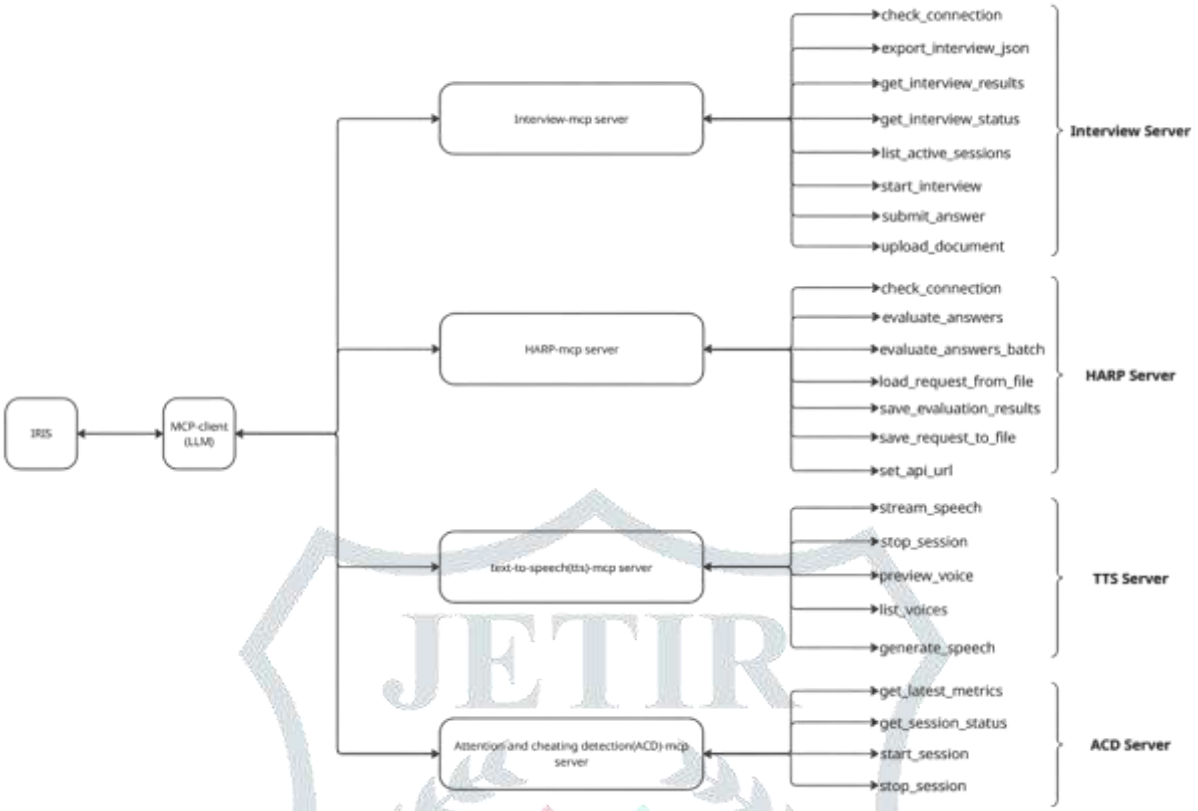

Figure 1: IRIS System Architecture showing component interactions and data flow

Table 1 details the component interaction in the IRIS system.

Component	***Input***	***Processing***	***Output***
Interview MCP Server	*User commands, Document uploads*	*Session management, Document handling*	*Interview sessions, Upload confirmations*
HARP MCP Server	*Candidate responses, Expected answers*	*Response evaluation, Scoring*	*Evaluation metrics, Feedback*
AllTalk TTS MCP Server	*Text, Voice/Language selection*	*TTS Request Formatting, API Call*	*Audio URL/Stream Confirmation, Voice List*
Attention and Detection MCP server	*Webcam Feed*	*CV Analysis (Gaze, Drowsiness)*	*Attention Metrics (JSON)*
Claude Interface	*User interaction*	*Natural language processing*	*Interview questions, Results presentation*

Table 1: Component Interaction in IRIS System

B. Interview Process Flow

The IRIS interview process follows a well-defined sequence, orchestrated by the Interviewer Host LLM interacting with the various backend services via MCP tools. The flow is designed to guide the candidate through a document-based interview, evaluate their responses, monitor their behavior, and present comprehensive results. Figure 2 illustrates this sequential flow.

1. Initialization and Document Upload:

- The process typically begins with the candidate (or administrator) providing the document (e.g., PDF) relevant to the interview.
- The Interviewer Host uses the upload_document tool (Interview Assistant MCP Server) to send the file to the Interview & RAG Service (/upload-document endpoint).

- The backend service saves the document and triggers the EmbeddingManager to process it, generating and caching embeddings if they don't already exist for this document hash. Confirmation is sent back to the host.

2. Starting the Interview:

- The candidate signals readiness to start.
- The Interviewer Host uses the start_interview tool (Interview Assistant MCP Server).
- The MCP server calls the backend (/interview/start/{session_id} endpoint), optionally specifying the document path.
- The backend initializes a new session, loads the appropriate RAG chain (using cached embeddings if available), and uses the RAG chain to generate the first interview question and its expected answer.
- The first question is returned via the MCP tool to the Interviewer Host.
- (Optional: Concurrently, an external trigger or the host could initiate behavior monitoring by calling the /start endpoint of the Attention Detection Service.)

3. Interview Loop (Question-Answer Cycle):

- The Interviewer Host presents the current question to the candidate.
- The candidate provides their answer verbally or textually.
- The Interviewer Host captures the candidate's answer.
- The Host uses the submit_answer tool (Interview Assistant MCP Server), providing the current session_id and the candidate's answer.
- The MCP server calls the backend (/interview/answer/{session_id} endpoint) with the answer.
- The backend stores the user's answer for the current question and checks if the interview question limit (e.g., 10 questions) has been reached.
- If not complete, the backend uses the session's RAG chain to generate the next question and expected answer.
- The next question is returned via the MCP tool to the Interviewer Host.
- The loop repeats from step 3a.

4. Interview Completion:

- Once the predefined number of questions is answered, the submit_answer tool call results in the backend returning an "Interview complete" message instead of a new question.
- The backend saves the complete interview data (all questions, RAG answers, user answers) for the session, typically to a JSON file (results/interview_{session_id}.json).
- The Interviewer Host informs the candidate that the question-answering phase is complete.

5. Evaluation Phase:

- Following the instructions in its system prompt, the Interviewer Host proceeds to the evaluation phase.
- It uses the get_interview_results tool (Interview Assistant MCP Server) to fetch the complete interview data (Q&A pairs) saved in step 4.
- The Host formats this data into the structure required by the HARP evaluation tool.
- It then invokes the evaluate_answers_batch tool (HARP MCP Server), passing the formatted interview data.
- The HARP MCP Server communicates with the HARP Evaluation Service backend, which performs the detailed LLM-based assessment of each answer against the question and the RAG-generated answer.
- The HARP service returns a structured JSON containing detailed scores (per parameter, overall) and explanations for each question.

6. Result Consolidation and Presentation:

- The HARP MCP Server relays the evaluation JSON back to the Interviewer Host.
- (Optional: The Host or another process triggers the `/stop` endpoint of the Attention Detection Service to get the final behavioral metrics JSON.
- The Interviewer Host now possesses the raw interview data, the detailed HARP evaluation scores, and potentially the behavioral metrics.
- Based on its system prompt, the Host synthesizes this information.
- It generates a final conclusion or report for the candidate/user, potentially including creating and displaying an artifact (e.g., graphs, metrics summaries) using specified UI elements to represent the results visually.

This structured flow ensures that the interview progresses logically, leveraging specialized backend services for document processing, question generation, and objective evaluation, all orchestrated by the central Interviewer Host LLM following predefined instructions and using MCP tools.

Figure 2 illustrates this process.

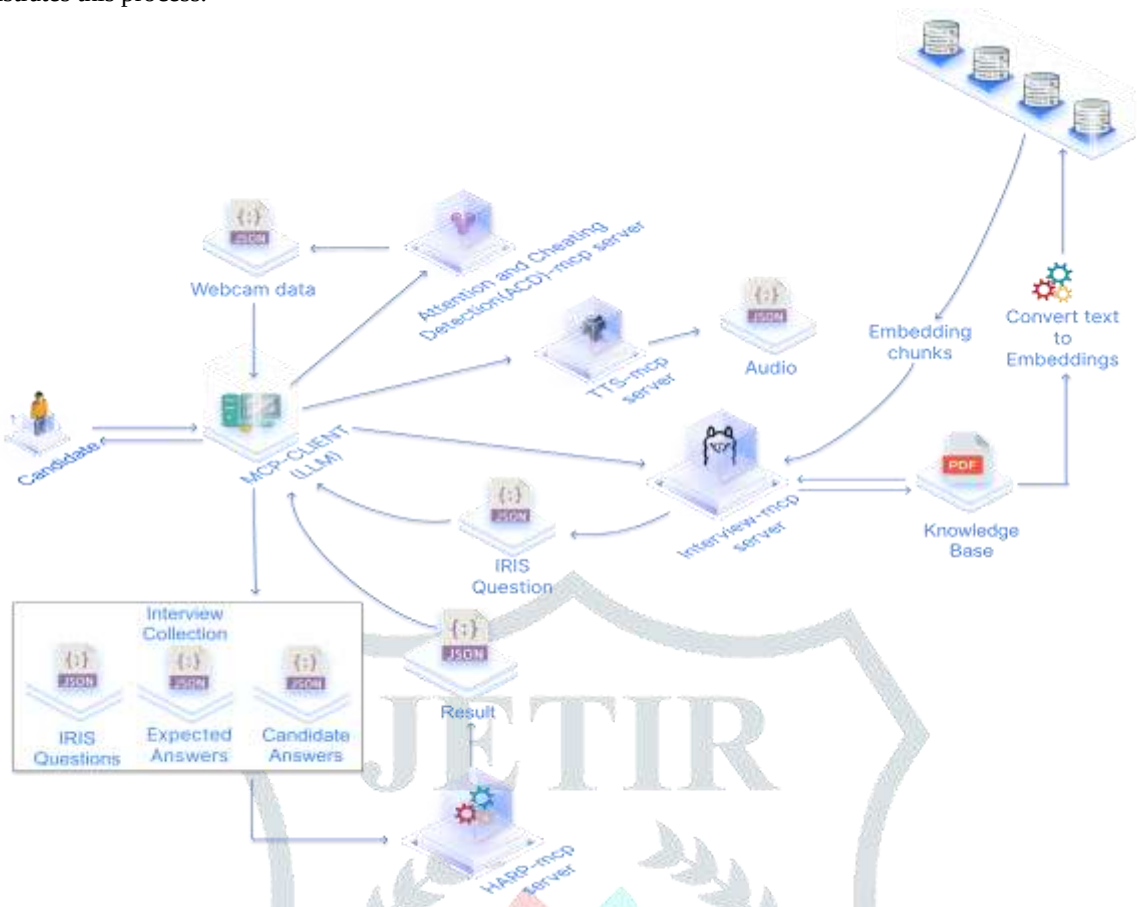

Figure 2: IRIS Interview Process Flow showing the sequence of operations from document upload to result presentation

IV. Core Technologies

A. Contextual RAG System

A cornerstone of IRIS is its advanced Retrieval-Augmented Generation (RAG) system, designed specifically for the interview context. Unlike standard RAG which primarily focuses on retrieving factual snippets, the IRIS Contextual RAG enhances relevance and coherence by preserving and utilizing document structure during retrieval and generation. This system is responsible for processing the source document, enabling context-aware retrieval, and generating relevant interview questions and expected answers. The implementation leverages the Langchain framework (Contextual RAG System Implementation).The ContextualHybridRetriever class combines vector-based semantic search with BM25 lexical search, followed by a neural reranking step. This hybrid approach leverages the strengths of both semantic and lexical search methods, resulting in more accurate and relevant context retrieval.

1. **Document Processing and Embedding:**

- **Ingestion:** Upon document upload (typically PDF, handled by PyPDFLoader), the text is extracted.
- **Chunking:** The document is segmented into manageable chunks using CharacterTextSplitter. This standard technique breaks down the text, but critically, the subsequent steps ensure context is not entirely lost.
- **Context Preservation & Addition:** A custom Context Builder step (implemented within the ContextualHybridRetriever) augments each chunk with metadata derived from the original document structure (e.g., page numbers, inferred section titles if available). This metadata is prepended to the chunk's content (e.g., "Page 5 | Section: Introduction: [chunk text]"), providing explicit context for the retrieval process.
- **Embedding:** The contextualized chunks are then embedded using a specialized sentence transformer model (Qwen Embeddings via EmbeddingManager). These embeddings capture the semantic meaning of the text along with its added context.
- **Vector Storage:** Embeddings are stored in a FAISS (Facebook AI Similarity Search) vector store, allowing for efficient semantic similarity searches (Contextual RAG System Implementation).
- **Caching:** An EmbeddingManager component handles the processing pipeline and implements caching (embedding_manager.py). It generates a hash of the input document and stores the processed embeddings and chunk data locally. Subsequent requests for the same document (identified by hash) reuse the cached data unless reprocessing is explicitly forced, significantly speeding up initialization for previously seen documents.

2. **Contextual Hybrid Retriever:**

- **Semantic Search:** Uses the FAISS vector store and Qwen embeddings to find chunks semantically similar to the query (e.g., a prompt asking for a question topic).
- **Lexical Search:** Utilizes a BM25Okapi algorithm (BM25 Engine) on the tokenized text of the contextualized chunks. BM25 is effective at matching keywords and provides a complementary retrieval signal to the semantic search.
- **Hybrid Combination:** Results from both semantic and lexical searches are combined and deduplicated.

- **Neural Reranking:** The combined set of candidate documents undergoes a reranking step. The retriever encodes the candidate chunk contents and the query using the Qwen embedding model. It calculates cosine similarity scores between the query embedding and each chunk embedding. The chunks are then re-sorted based on this relevance score, prioritizing the most contextually pertinent information. This step refines the results beyond simple similarity or keyword matching (Contextual RAG System Implementation). The top-ranked documents (typically limited to around 10 for efficiency) are passed to the generation step.

3. **Question and Answer Generation:**

- **LLM:** The generation component uses a powerful Large Language Model accessed via an API.
- **Prompting:** A specialized prompt template (qa_prompt) instructs the LLM to act as an interviewer and generate challenging questions based solely on the retrieved context provided by the ContextualHybridRetriever. The prompt also requests the LLM to generate an expected answer for the question it creates.
- **Chain Execution:** The entire process is orchestrated using a Langchain Runnable chain (rag_chain). A query (e.g., "Generate the next interview question") triggers the ContextualHybridRetriever to fetch relevant context, which is then combined with the question and passed to the LLM via the qa_prompt for final question and answer generation.

Figure 3 shows the contextual RAG system workflow.

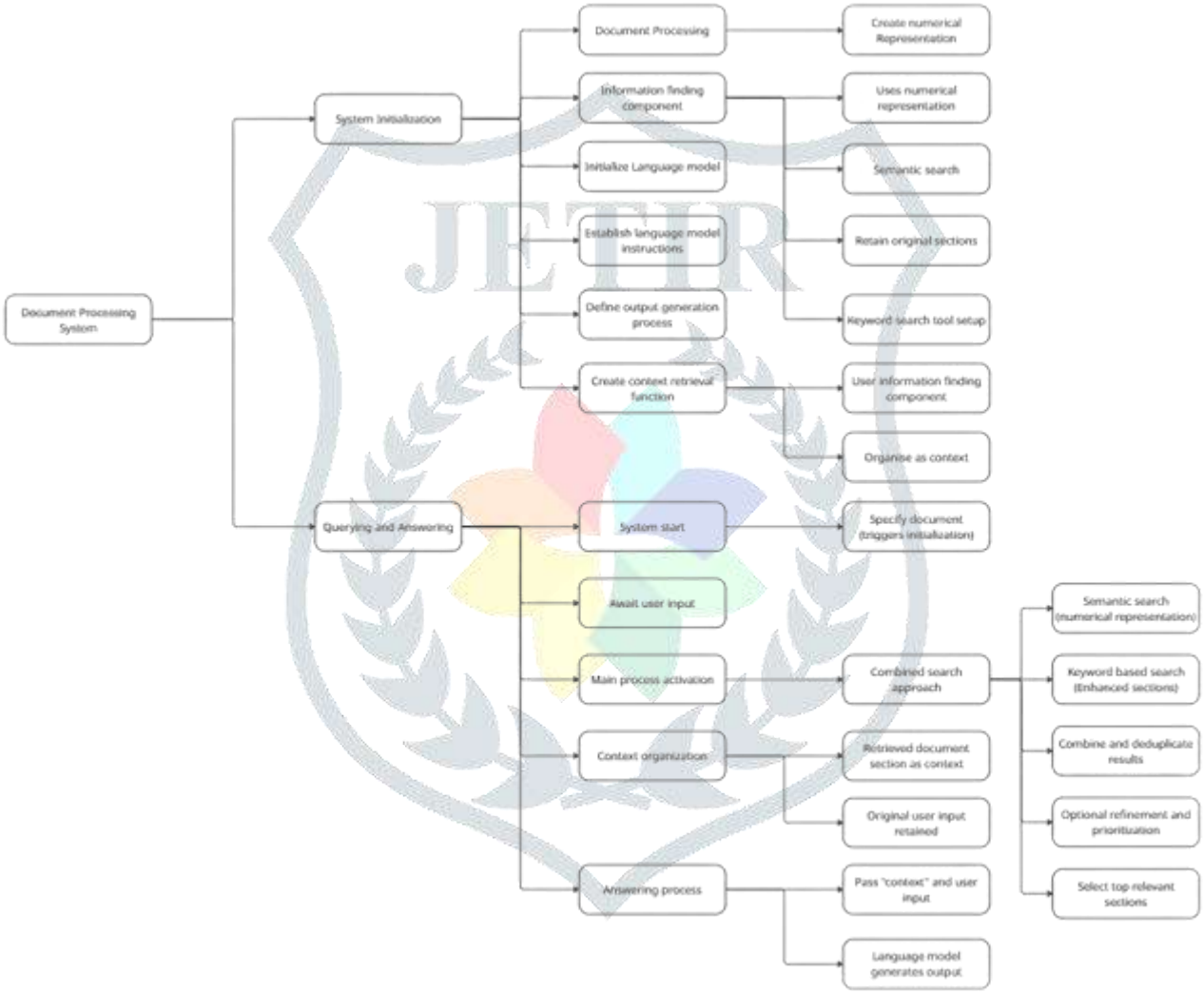

Figure 3: Contextual RAG System Workflow illustrating the process from document ingestion to response generation

This multi-stage Contextual RAG system ensures that generated questions are not only relevant to the document's content but also grounded in specific sections or pages, facilitated by the context preservation and hybrid retrieval strategy. The neural reranking further refines the selection, leading to more focused and pertinent interview interactions.

Table 2 outlines the components of the contextual RAG system and their functions.

Component	**Function**	**Implementation**
PDF Loader	Extract text from PDF documents	PyPDFLoader

Text Splitter	Split document into manageable chunks	CharacterTextSplitter
Context Builder	Add document context to chunks	Custom implementation
Embedding Model	Generate vector representations	QwenEmbeddings
Vector Store	Store and retrieve embeddings	FAISS
BM25 Engine	Perform lexical search	BM25Okapi
Reranker	Score and reorder retrieved documents	Custom neural implementation

Table 2: Contextual RAG Components and Functions

B. HARP Evaluation System

To ensure objective and consistent assessment of candidate responses, IRIS incorporates the HARP (Human Answer Relevance Protocol) evaluation system. HARP moves beyond simple keyword matching or subjective human judgment by leveraging LLMs to perform nuanced evaluations based on predefined criteria aligned with the interview's context and the source document.

1. Core Evaluation Service:

- **HARP Evaluation Service (FastAPI Backend):** At the heart of the system is a dedicated FastAPI microservice. This service exposes an API endpoint (primarily /evaluate/) designed to receive evaluation requests in batches.
- **LLM-Powered Evaluation:** The core evaluation logic resides within a LlamaSimilarityEvaluator component. This component utilizes an external LLM API (e.g., Llama models) to compare the candidate's answer (candidate) against both the original interview question (question) and the RAG-generated expected answer (rag). This three-way comparison allows for assessing not just similarity to the expected answer but also direct relevance to the question asked.
- **Multi-Dimensional Assessment:** HARP performs a multi-dimensional evaluation based on several key parameters, typically including:
  - **Factual Accuracy:** Consistency with the source document content (implicitly checked by comparing to the RAG-generated answer).
  - **Relevance:** How directly the candidate's answer addresses the question asked.
  - **Completeness:** Coverage of the key points expected in the answer.
  - **Clarity:** How clear and understandable the response is.
  - **Conciseness:** Efficiency of expression, avoiding unnecessary verbosity.

- **Customization and Strictness:** The system allows for flexibility. Default parameters and their descriptions are used, but evaluation requests can optionally include custom parameters or specify different strictness levels (e.g., "low", "medium", "high") for each parameter. This allows tailoring the evaluation focus for different interview types or questions.
- **Scoring Formats:** HARP supports multiple scoring formats, including "0-10", "0-5", and "percentage", which can be specified per evaluation request. It calculates scores for individual parameters and derives an overall_score.

2. Invocation via MCP:

- **HARP MCP Server (HARP-Evaluator):** Direct interaction with the HARP Evaluation Service is handled by the HARP MCP Server. This server acts as the intermediary between the Interviewer Host LLM and the evaluation backend.
- **Evaluation Tools:** It provides MCP tools, primarily evaluate_answers (for single evaluations) and evaluate_answers_batch (for evaluating multiple Q&A pairs efficiently).
- **Workflow Integration:** As defined in the interview flow, after the interview interaction phase is complete, the Interviewer Host LLM is instructed to:
  - Retrieve the full set of interview questions, RAG-generated answers, and the candidate's answers using the get_interview_results tool from the Interview Assistant MCP server.
  - Format this data as required by the evaluate_answers_batch tool.
  - Invoke the evaluate_answers_batch tool on the HARP MCP Server, passing the collected interview data.
  - The HARP MCP Server then calls the HARP Evaluation Service API (/evaluate/) with this data.
- **Results Handling:** The HARP Evaluation Service processes the batch evaluation and returns detailed JSON results (including individual parameter scores, explanations, and overall scores) to the HARP MCP Server. The MCP server relays this structured JSON data back to the Interviewer Host LLM. The LLM is then responsible for interpreting this data and potentially generating a final report or artifact for the user, including visualizations as requested.

By decoupling the evaluation logic into a dedicated service and providing access via MCP tools, HARP ensures that the Interviewer Host can orchestrate objective, configurable, and detailed evaluations without needing to perform the complex assessment itself. This maintains consistency and leverages the analytical capabilities of LLMs for robust response analysis. Figure 4 should depict this flow, showing the Interviewer Host using the HARP MCP Server to trigger evaluations in the HARP backend service.

Figure 4 shows the HARP system workflow.

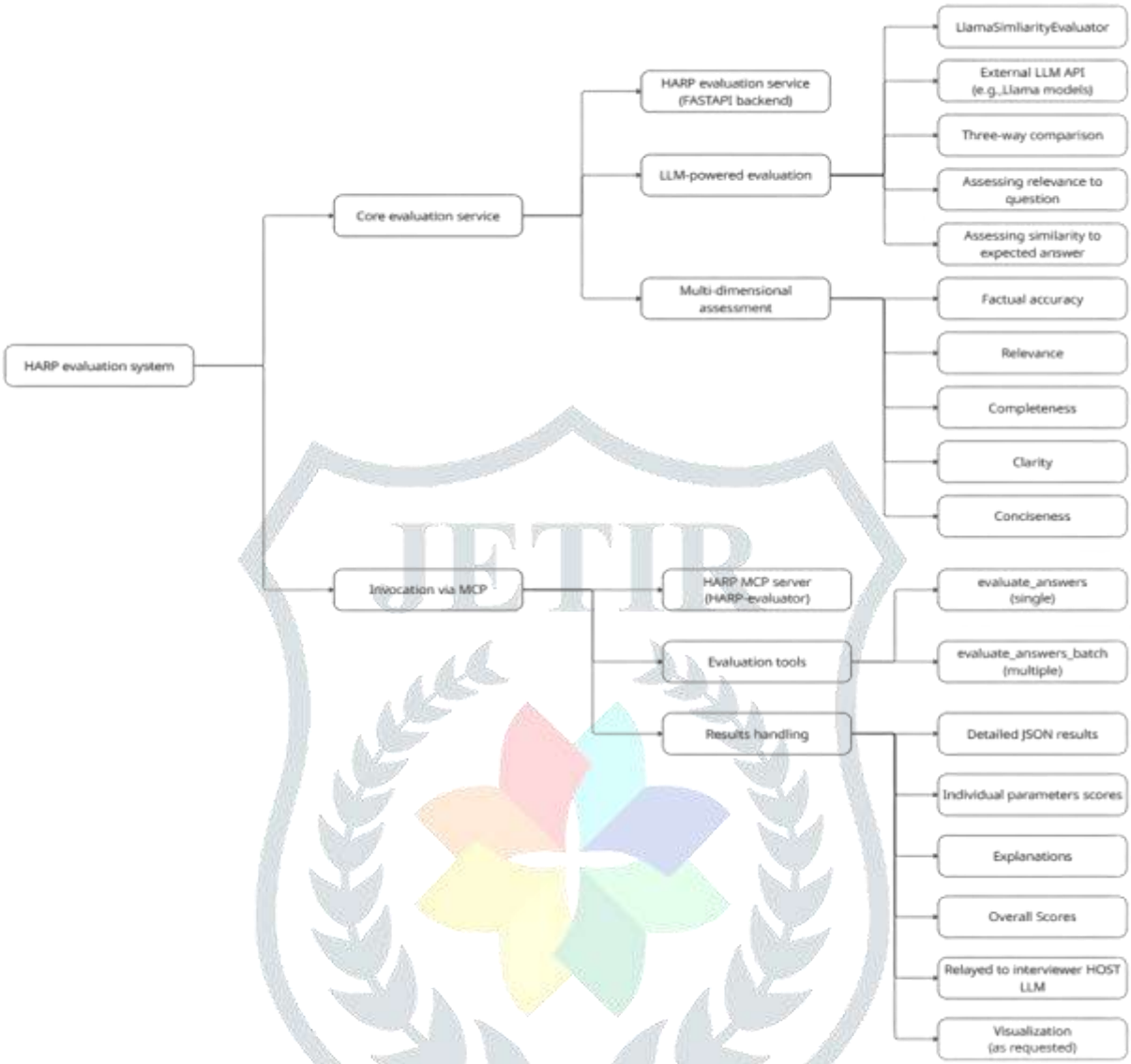

Figure 4: HARP system Workflow illustrating all the components responsible for result evaluation

Table 3 presents the evaluation parameters used in HARP.

Parameter	**Description**	**Strictness Levels**
Factual Accuracy	Correctness of information	Low, Medium, High
Relevance	Pertinence to the question	Low, Medium, High
Completeness	Coverage of key points	Low, Medium, High
Conciseness	Efficiency of expression	Low, Medium, High
Clarity	Clear communication	Low, Medium, High

Table 3: HARP Evaluation Parameters

C. Text-to-speech Integration (AllTalk/XTTS):

To enhance the interactive nature of the interview simulation, IRIS incorporates a Text-to-Speech (TTS) module facilitated by the `AllTalk TTS MCP Server`. This component allows the Interviewer Host LLM to vocalize its outputs, such as questions and feedback.

1. **MCP Server Interface:** The `AllTalk TTS MCP Server` acts as an intermediary, providing tools callable by the Interviewer Host. Key tools include `generate_speech` (generates audio file), `stream_speech` (real-time streaming), `list_voices`, and `preview_voice`.
2. **Backend TTS Engine:** The MCP server communicates with a separate AllTalk TTS API backend service utilizing the XTTS model (a high-quality multilingual voice cloning TTS engine) to perform the actual conversion of text to audible speech.
3. **Integration:** The Interviewer Host invokes the MCP tools (`generate_speech` or `stream_speech`) to speak its responses instead of only displaying text, adding a layer of auditory interaction configured via the AllTalk system (voices, languages).

V. Behavior Monitoring using Computer Vision

Beyond evaluating the content of responses, IRIS incorporates a computer vision module for real-time monitoring of candidate behaviour during the interview. This component aims to enhance interview integrity by assessing candidate attention levels and detecting potential indicators of distraction or disengagement. The insights derived from this module are designed to contribute to the overall candidate assessment (IRIS System Overview Document).

A. Core Technologies And Processing Pipeline

The behavior monitoring system operates as an independent FastAPI service that processes the candidate's webcam feed. Key technologies employed include:

1. **Video Capture:** Accesses the candidate's webcam (cv2.VideoCapture(0)) to capture video frames in real-time.
2. **Facial Landmark Detection:** Utilizes Mediapipe's Face Mesh (mp.solutions.face_mesh) to detect detailed facial landmarks (eyes, mouth, face contour) on each frame. These landmarks are crucial inputs for subsequent analyses.
3. **Gaze Estimation:** Implements the L2CS-Net model (gaze_detection.l2cs_net.l2cs.Pipeline) to estimate the candidate's gaze direction (pitch and yaw) based on the detected facial features. This helps determine if the candidate is looking at the screen or significantly away.
4. **Drowsiness Detection:** Integrates a dedicated module (drowsiness_detection) that analyzes facial landmarks, likely focusing on the eye aspect ratio (EAR) derived from Mediapipe landmarks, to detect signs of drowsiness (e.g., prolonged eye closure, yawning).
5. **Distraction Logic:** A core function synthesizes information from gaze estimation and drowsiness detection to determine if a frame exhibits distraction. Gaze deviation beyond a certain threshold or detection of drowsiness/sleeping states typically flags a frame as 'distracted'. The system tracks the cumulative count of distracted frames. While the broader IRIS concept mentions YOLOv8 for detecting potential cheating behaviors like using unauthorized resources (IRIS System Overview Document), the attentiondetection mechanism focuses specifically on monitoring attention through gaze and drowsiness.

B. Monitored Metrics

The service continuously tracks and calculates several quantitative metrics throughout the monitoring session:

1. **Frame Counts:** total_frames, drowsy_frames (frames where drowsiness/sleep was detected), gaze_distracted_frames (frames where gaze deviated significantly), overall_distracted_frames (total frames flagged as distracted by any criteria).
2. **Percentage Metrics:**
- **drowsy_percentage:** (drowsy_frames / total_frames) * 100
- **gaze_distracted_percentage:** (gaze_distracted_frames / total_frames) * 100
- **overall_distracted_percentage:** (overall_distracted_frames / total_frames) * 100
- **attention_percentage:** 100 - overall_distracted_percentage (representing the proportion of time the candidate was deemed attentive).

3. **Time Metrics:**
- **elapsed_time_seconds:** Total duration of the monitoring session.
- **distracted_time_seconds:** Estimated time the candidate was distracted (overall_distracted_percentage * elapsed_time_seconds).
- **attention_time_seconds:** Estimated time the candidate was attentive (elapsed_time_seconds - distracted_time_seconds).

4. These metrics are updated periodically during the session and logged (log_student_behaviour.csv).

C. Integration and Influence on Results

1. **API Control:** The monitoring process is controlled via the FastAPI service's endpoints. A /start request initiates the background video capture and analysis thread for a specific session ID. A /stop request signals the thread to terminate, calculates the final metrics, saves them to a JSON file (e.g., session_{session_id}_final.json), and returns this final JSON payload. A /status endpoint allows querying the latest metrics during an active session.
2. **Impact on Evaluation:** The design specifies that the outputs of this behavior monitoring module influence the candidate's overall result (IRIS System Overview Document). The final metrics JSON returned by the /stop endpoint (containing attention_percentage, distracted_time_seconds, etc.) is retrieved by the Interviewer Host or a subsequent processing step. This

quantitative behavioral data can then be factored into the final assessment alongside the HARP evaluation scores, potentially adjusting the overall score, flagging inconsistencies, or providing additional context for human review.

This integration provides a more holistic view of candidate engagement, adding a layer of behavioral assessment to the content-based evaluation performed by HARP. Figure 5 should illustrate the components of this module: Webcam Input -> Mediapipe -> L2CS-Net / Drowsiness Detection -> Distraction Logic -> Metrics Calculation -> API Output.

Figure 5 illustrates the YOLO v8 integration for behavior monitoring.

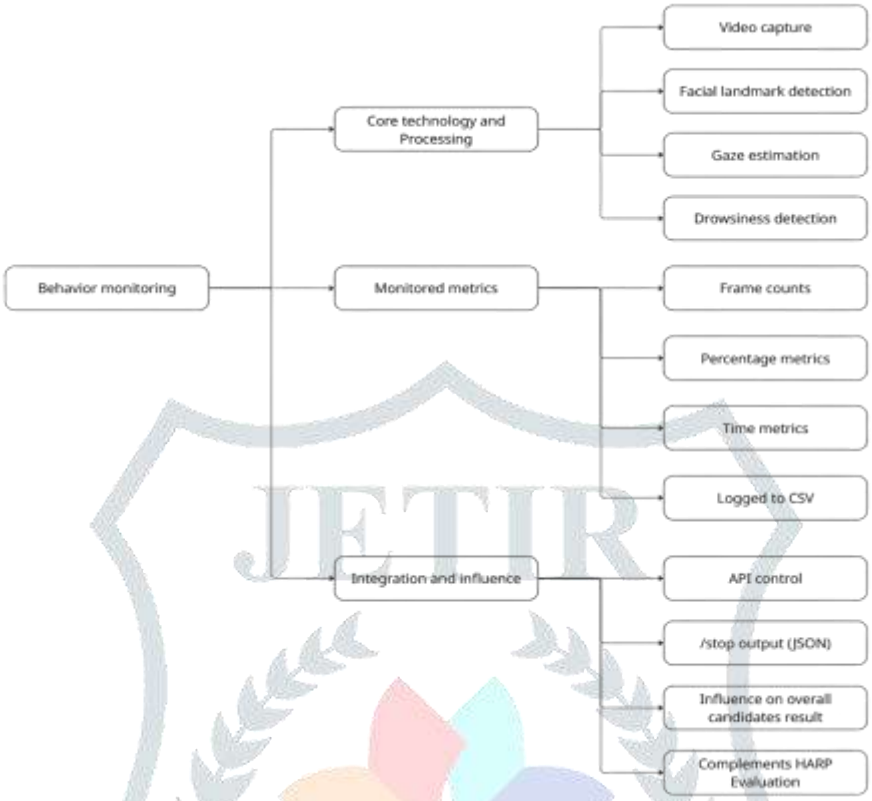

Figure 5: YOLO v8 Integration for Behavior Monitoring showing object detection and attention tracking

VI. Experimental Evaluation

A. Methodology

To evaluate the effectiveness of IRIS, we conducted experiments comparing it to traditional interview methods across multiple dimensions. The evaluation was performed with 50 participants from diverse backgrounds, using standardized documents and evaluation criteria.

Participants were randomly assigned to two groups:

1. **Control Group:** Interviewed by human interviewers using traditional methods
2. **IRIS Group:** Interviewed by the IRIS system

The evaluation metrics included:

1. Question relevance and quality
2. Response evaluation consistency
3. Interview time efficiency
4. Participant satisfaction
5. Cheating detection accuracy

B. Results

The experimental results demonstrate significant improvements in several key metrics when using IRIS compared to traditional methods.

Table 4 summarizes these results.

Metric	Traditional Method	IRIS	Improvement
Question Relevance (1-10)	6.8	8.7	+27.9%
Evaluation Consistency (1-10)	5.9	9.2	+55.9%
Time Efficiency (minutes/interview)	45.3	18.7	+58.7%

Participant Satisfaction (1-10)	7.1	8.3	+16.9%
Cheating Detection Accuracy	43%	92%	+114.0%

Table 4: Experimental Results Comparing Traditional Methods to IRIS

The results indicate that IRIS significantly outperforms traditional methods in all evaluated metrics. Particularly notable is the improvement in evaluation consistency, which addresses one of the key challenges in traditional interview processes.

Figure 5 presents the distribution of scores for question relevance and evaluation consistency.

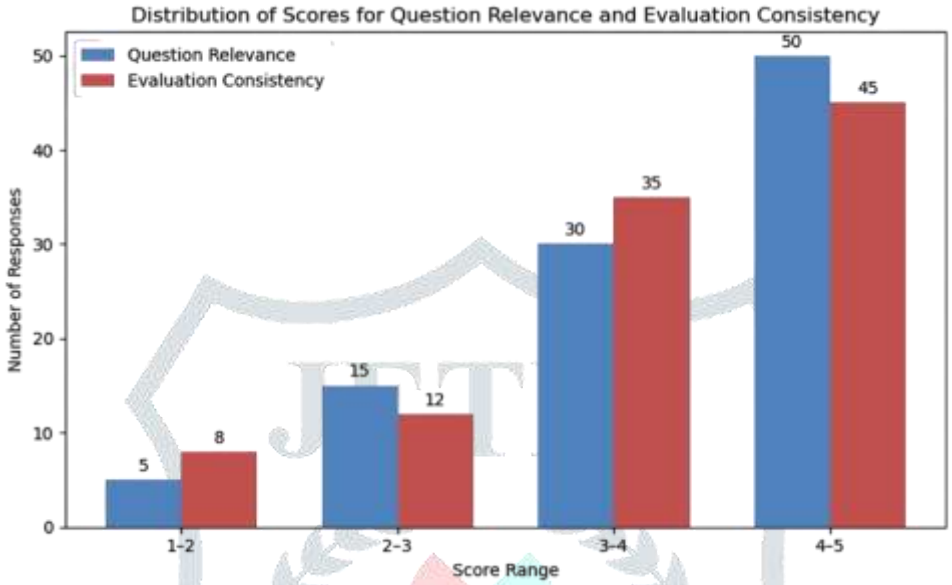

Figure 5: Distribution of Scores for Question Relevance and Evaluation Consistency

C. Case Studies

To provide qualitative insights into the system's performance, we conducted three case studies in different domains:

1. **Academic Assessment:** Using textbook materials to evaluate student understanding
2. **Technical Interview:** Using technical documentation for programmer interviews
3. **Research Interview:** Using research papers for domain expert interviews

In all three cases, IRIS demonstrated adaptability to domain-specific requirements while maintaining high performance. The system was particularly effective in the technical interview case, where precision in question generation and response evaluation is crucial.

VII. Discussion and Future Work

A. Limitations

While IRIS demonstrates significant improvements over traditional methods, several limitations should be acknowledged:

1. **Document Format Limitations:** The current implementation supports only PDF documents, limiting applicability in environments with diverse document formats
2. **Language Constraints:** The system is optimized for English, with limited support for other languages
3. **Hardware Requirements:** The computer vision component requires a webcam and substantial computational resources
4. **Contextual Understanding:** Despite advances in RAG, the system may still miss subtle contextual nuances in highly specialized documents
5. **TTS Related:** Potential latency; voice quality affecting perception; dependency on TTS backend.

B. Future work

Based on the identified limitations and user feedback, several directions for future work are planned:

1. **Expanded Document Support:** Implementing support for additional document formats such as DOCX, HTML, and markdown
2. **Multilingual Capabilities:** Enhancing the system to handle interviews in multiple languages
3. **Integration with Video Platforms:** Developing connectors for popular video conferencing tools to enable remote interviews
4. **Adaptive Questioning:** Implementing dynamic question generation that adapts to candidate responses
5. **Privacy Enhancements:** Strengthening privacy protections for sensitive interview data
6. **TTS Enhancements:** More expressive/emotional TTS; lower latency streaming

Table 5 outlines the proposed future enhancements.

Enhancement	Description	Benefit	Complexity
Multiple Document Formats	Support for DOCX, TXT, HTML	Increased flexibility	Medium
Multilingual Support	Interview in multiple languages	Global applicability	High
Video Integration	Connection with video platforms	Remote interview capability	Medium
Custom Parameters	User-defined evaluation metrics	Tailored assessment	Low
Adaptive Questioning	Dynamic question adjustment	Personalized interviews	High

Table 5: Proposed Future Enhancements

VIII. Conclusion

This paper introduced IRIS, an AI-powered system integrating contextual RAG, automated HARP evaluation, behavior monitoring, and TTS vocalization for conducting enhanced document-based interviews. By leveraging an MCP-orchestrated modular architecture, IRIS addresses key challenges in traditional interview processes, including consistency, objectivity, efficiency, and interactivity. Experimental results demonstrate significant improvements across multiple metrics compared to traditional methods. The system's adaptability suggests potential for transforming how document-based assessments are conducted across educational, professional, and research domains, offering a more standardized, objective, efficient, and engaging interview experience.

IX. Acknowledgment

We would like to thank Prof. Sheela Verma, our supervisor, for their invaluable guidance, continuous encouragement, and expert advice throughout the course of this project. Their expertise in Computer Vision, Image Processing, ML, and AI greatly enhanced our research. We also acknowledge the Department of Computer Science of Bhilai Institute of Technology, Raipur, providing the necessary resources and infrastructure to carry out this research.

References

[1] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877-1901.

[2] Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLO by Ultralytics. GitHub repository. https://github.com/ultralytics/ultralytics.

[3] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33, 9459-9474.

[4] Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. W. (2020). Retrieval augmented language model pre-training. International Conference on Machine Learning, 3929-3938.

[5] Khattab, O., & Zaharia, M. (2020). ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 39-48.

[6] Kumar, R., Chaudhary, K., & Thille, C. (2020). Scaling automated feedback for open-response assignments. Proceedings of the Seventh ACM Conference on Learning @ Scale, 331-334.

[7] Zhao, Z., Hoang, L., Joshi, A., Chen, X., & Wang, W. Y. (2021). Good examples make a faster learner: Simple demonstration-based learning for low-resource NER. arXiv preprint arXiv:2110.08454.

[8] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., & Zhou, D. (2023). Self-consistency improves chain of thought reasoning in language models. International Conference on Learning Representations.

[9] Pei, L., & Zhong, Y. (2022). Eye tracking for assessment: A systematic literature review. Educational Psychology Review, 34(1), 387-426.

[10] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 779-788.

[11] Chase, H. (2022). Langchain: Building applications with LLMs through composability. GitHub repository. https://github.com/hwchase17/langchain.

[12] Ramírez, S. (2018). FastAPI: Modern, Fast Web Framework for Python. GitHub repository. https://github.com/tiangolo/fastapi.
[13] Anthropic. (2023). Model Context Protocol: Tool Use Framework for LLMs. Technical Report.
[14] Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. Foundations and Trends in Information Retrieval, 3(4), 333-389.