**AI Mock Interviewer for Data Analytics/Science**

Team 1

Gnana Prasuna Nimeesha Vakacharla, Kalyani Zope, Naga Sai Sindhura Pandrangi, Sonia Bathla

Department of Applied Data Analytics

DATA 298-A : Final Project Report

DR. Simon Shim

May 12, 2025

**Abstract**

InterviewPrep-AI is an AI-powered web-based platform that will transform Mock interview preparation through tailored, real-time feedback for Data Analytics and Data Science profiles. Because traditional interviewing methods do not provide corrective feedback in real time, generate questions specifically for the candidate, or adapt to different formats. As AI-powered job interviews become ever-increasingly common place, the need for smart, interactive mock-job interview platforms to better prepare candidates is growing. This project will fill this gap, by generating realistic interviews on a wide variety of formats:  technical coding, data structures & algorithms (DSA), behavioral (STAR based), case study (Business, Analytical Problem-solving), and HR (soft skills, leadership). It uses an AI-based interview flow and users can then optionally add resumes and job descriptions and select their role and the mode in which they want to respond (text, audio, or video) for it to generate AI-based job-specific questions. The responses will be analyzed using Deep Learning & NLP-based evaluation and assess it for accuracy, clarity & problem-solving skills and give qualitative feedback instantaneously, along with personalized suggestions to improve. The top-end technologies will make the solution work: Next. js & Tailwind CSS for a seamless UI Node. js & Express. js (for building a scalable API) and MongoDB/PostgreSQL (for secure data storage). So, an LLM-powered AI Model allows for asking dynamic questions based on the context and evaluating the output, whereas RAG improves the analysis of responses. Cloud-based deployment (AWS /Vercel), Docker & Kubernetes-based scalabilities, Database-MongoDB Atlas for data management. This will help job seekers through realistic, AI-led interview simulations, assisting recruiters in evaluating candidates, aiding the academia by integrating with career coaching and boot camps, and enabling corporate training by connecting with employee skill evaluation. This InterviewPrep-AI is expected to change the hiring and career development landscape by making interview preparation an interactive, adaptive, and extremely scalable process.

## **Index**

1.Project Background and Executive Summary………….………………………………………….4   1.1 Project Background……………………………..…………………………………………..4  
   1.2 Project Requirement…………………………..……………………………........................6  
   1.3 Project Deliverables……………………………….……………………………………….7  
   1.4 Technology and Solution Survey……………………………….………………………….8  
   1.5 Literature Survey of Existing Research……………………….………………………….10

2.Data and Project Management Plan……………………..………………………………………..14  
   3.1 Data Management Plan……………….…………………………………………………..14  
   3.2 Project Development Methodology…….………………………………………………...16  
   3.3 Project Organization Plan……………….………………………………………………..17  
   3.4 Project Resource Requirements and Plan……………….………………………………..19  
   3.5 Project Schedule…………..………………………………………………………………23

3.Data Engineering…………………………………………………………………………………………25  
   4.1 Data Process…..…………………………………………………………………………..25   4.2 Data Collection…………………………………………………………………………... 26

  4.3 Data Preprocessing………………………………………………………………………..28  
   4.4 Data Transformation……………………………………………………………………...30  
   4.5 Data Preparation………………………………………………………………………….35  
   4.6 Data Statistics…………………………………………………………………………….39

4.Model Development………………………………………………………………………………50  
   5.1 Model Proposal…………………………………………………………………….……..50  
   5.2 Model Supports……………………………………………………………..…………….54

  5.3 Model Comparison and Justification……………………………………………………..58  
   5.4 Model Evaluation Methods……………………………………………………..………...61

5.References……………………………………..………………………………………………….63

## **List of Figures**

## Figure 2.1 Gantt Chart…………………………………………………………………………….23 Figure 2.2 PERT Chart……………………………………………………………………………24 Figure 3.1 DSA Questions Raw Sample Dataset…………………………………………………27 Figure 3.2 Technical Questions Raw Sample Dataset………………………………………….....27 Figure 3.3 SQL Questions Raw Sample Dataset………………………………………………….28 Figure 3.4 DSA Questions Pre-processed Sample Dataset……………………………………….29 Figure 3.5 Technical Questions Pre-processed Sample Dataset…………………………………..30 Figure 3.6 SQL Questions Pre-processed Sample Dataset………………………………………..30 Figure 3.7 List of Composers……………………………………………………………………..33 Figure 3.8 DAG Scripts for Automation of Data Pipeline………………………………………..33 Figure 3.9 Tech Questions Dataset after Transformation and Feature Engineering……………...34 Figure 3.10 StrataScratch Questions Dataset after Transformation and Feature Engineering…....34 Figure 3.11 LeetCode Questions Dataset after Transformation and Feature Engineering……….34 Figure 3.12 Train, Test, and Validation Dataset for StrataScratch………………………………..37 Figure 3.13 Train, Test, and Validation Dataset for LeetCode………………………………...….38 Figure 3.14 Train, Test, and Validation Dataset for Tech Questions……………………..…….…38

## Figure 3.15 Raw, Pre-processed, and Transformed Tech Questions Dataset………………..……40 Figure 3.16 Summary of Data after Preprocess and Transformation…………………………….41 Figure 3.17 Split of the Dataset and Visualizations…………………………………………...….42

## Figure 3.18 Raw, Pre-processed and Transformed Strata Scratch Dataset…………………….…44 Figure 3.19 Summary of the Data after Preprocess and Transformation……………………..…..45 Figure 3.20 Split of the Dataset and Visualizations…………………………………………...….46

## Figure 3.21 Raw, Pre-processed and Transformed Leetcode Dataset………………………….…47 Figure 3.22 Summary of the Data after Preprocess and Transformation…………………………48 Figure 3.23 Split of the Dataset and Visualizations………………………………………………49 Figure 4.1 AI Mock Interviewer Platform Architecture………………………………………….57

## Figure 4.2 Comparison of baseline and fined tuned Model………………………………………..61

## **List of Tables**

Table 1.1 Comparison Table: InterviewPrep-AI vs. Existing Solutions…………………………..9

Table 1.2 Literature Survey Summary Table……………………………………………………..13

Table 2.1 Data Management Tasks and Owners………………………………………………….16

Table 2.2 Tools Used, Usage, and Cost…………………………………………………………...21

Table 4.1 Model Comparison………………………………………………………………..…....58

Table 4.2 Evaluation Metrics for Implemented Model…………………………………………...61

Table 4.3 Planned Evaluation Metrics for Proposed Models……………………………………..62

## **1. Project Background and Executive Summary**

### **1.1 Project Background**

The job market is highly competitive for data professionals. Even though data professionals are in high demand these days, several candidates face systematic interview preparation and experience interview anxiety. Regular mock interviews are narrow in perspective, designed for few feedback queries and never personalized.

### **Needs and Importance**

AI-Centric Mock Interview Platforms- These mock interview platforms allow candidates to practice practically and address the shortcomings of a traditional mock interview approach by generating interview experience based on AI algorithms that analyze the candidate’s experience, skills, role of the job, etc. Instant feedback on accuracy, relevance, and problem-solving skills contextualizes and reinforces the candidate’s preparation and increases their confidence when it goes to the real-world interviews.

### ***Targeted Problem:***

* + - Customized and dynamic interview preparation for Data Analyst and Data Scientist roles available.
    - Existing mock interviewing platforms, which provide limited real-time feedback and evaluation of performance.
    - A solution that is scalable and adaptable for continuous learning and improvement.

### ***Motivations and Goals***

* + - To provide a scalable, AI-based solution enabling custom mock interviews by job role and experi- ence.
    - Provide immediate feedback and tailored improvement suggestions.
    - Use real-world complex interview questions to prepare candidates and raise their success rates in those interviews.

### **Planned Project Approach and Method**

### ***Approach:***

The project utilizes a combination of AI, web, and cloud infrastructure:

* + - **AI Integration:** RAG for response analysis and dynamic question generation using LLMs.
    - **User-Centric Design:** Easy to use front end with choices for responses (Text, Audio, or Video), ability to upload resume and job description to facilitate responses.
    - **Scalable Architecture:** Deploy using AWS or Vercel on the cloud for availability and scalability.

### ***Methods:***

* + - Collect and analyze a vast amount of job descriptions and resumes to train the AI for context-aware question generation.
    - Use Deep Learning and Natural Language Processing to score the response in terms of accuracy, clarity, and problem-solving skills.
    - Use the agile development methodology to iteratively design and test the platform, incorporating user feedback to further refine the functionality.

### **Expected Contributions and Applications *Contributions:***

* + - An AI-driven platform that revolutionizes interview preparation, providing specific, real-time feed- back.
    - A better candidate experience with increased employability in terms of technical, behavioral, and HR interviews.
    - Scalable solutions that naturally plug into academic, corporate, and personal career development ecosystems.

### ***Applications:***

* + - **Job Seekers:** Maintain continuous practice to develop confidence and skills required to face the real interviews.
    - **Recruiters:** Assess candidates and optimize the hiring process.
    - **Academia:** Helps integrate production with career coaching and boot camps.
    - **Corporate Training:** Assessing employee skills and leadership development

## **1.2 Project Requirement**

### ***Functional Requirements:***

* + - User registration and login.
    - Upload a resume and a job description.
    - Generating dynamic questions of different levels depends on job role and experience.
    - Support for various response types (text, audio, video).
    - Evaluating responses driven by AI (generates correctness, clarity, problem-solving).
    - Real-time feedback with suggestions for improvement.
    - Tracking performance and history of previous interviews.

### ***AI-Powered Feature Requirements:***

* + - **Dynamic Question Adaptation:** The AI should adapt questions to user experience level with at least 90% of accuracy in difficulty calibration.
    - **Response Evaluation:** Use NLP and Deep Learning based models to evaluate the correctness and relevance of response and aim for at least 85% accuracy.
    - **Personalization:** Should be able to personalize the questions based on resumes being upload- ed/job descriptions provided by user with at least 80% specificity.

### ***Data Requirements:***

* + - **Input Data:** User-fetched resumes and job descriptions in PDF/text formats, role selected by the user, and recording responses in text/audio/video.
    - **Training Data:** Thousands of sample Interview Questions and Answers based on Coding, DSA, Behavioral, Case Study, Conceptual, and HR questions which are sourced from public reposito- ries,websites, and expert curated content.
    - **Storage Needs:** Secure storage for user data and AI model outputs, which will be AI-generated questions, evaluation scores, feedback for user review, and platform improvement.

## **1.3 Project Deliverables**

Our project is all about researching and developing an AI mock interviewer intended for Data Analytics, Data Science, and Programming, with reports, prototypes, development applications, and production applications being some of the deliverables. The aim is to help the students to practice and improve their technical and behavioral skills by using a strong AI-based tool that can simulate real-world interview scenarios. One of the major tasks is the utilization of Interviews by AI to inculcate them with such data science, analytics, machine learning, etc. and problem-solving skills. Furthermore, users preparing for behavioral questions will learn how to do project teamwork and will participate in the past project experiences, as well as they will reinforce their technical skills in SQL, Python, and data engineering concepts. The AI mock interviewer will give input to the user to help them improve their performance incrementally. In order to ensure quality, we will develop metrics to assess Large Language Models (LLMs) that drive the AI interviewer, which will guarantee accuracy, relevance, and effectiveness in responses. To add value, we will also be gathering domain-specific data sets of which we will use in the process of training and testing. This will ensure that the AI answers the questions of many different interviewers accordingly. The project’s outcome will be an extremely effective research and development process that will end in a fully operational machine that will imitate interviewers for Data Analytics, Data Science, and Programming. The reports will present the development process, tools application, and evaluation metrics for LLM that will provide a clear picture of the performance of the system and give hints for further enhancement. The first thing that will be made in the shape of prototypes will be the AIs holdings in the form of interviewers, so the usability of the product can be evaluated, and the creators can make subsequent amendments based on the performance scores and user feedback. Development applications will center around the implementation of the basic algorithms and interfaces to ensure the smooth interaction of the tools with the users and the design of an intuitive interface. Finally, the production applications will consist of a complete AI mock interviewer which will be able to serve as a software for deployment and personal use by any person preparing for interviews related to data technologies. The users’ face-to-face solution will be the tool that will let them get better at their interview skills, build up their self-confidence, and thus increase their chances of landing a job in a highly competitive job market.

## **1.4 Technology and Solution Survey**

### ***Existing Solutions:***

* + - * Platforms like Pramp and Interviewing.io give peer or expert input in the mock interviews, but no live AI analysis.
      * HireVue uses AI to analyze candidates in interviews, though not during mock preparation.
      * Generic AI systems and Google Dialog Flow may mimic dialogues, but are not intended for job-screening purposes.

### ***Feature Comparison:***

* + - * **Question Generation:** InterviewPrep-AI makes a distinction based on its ability to generate job-specific questions using resumes + job descriptions (LLMs), where most other platforms sell a static question bank.
      * **Feedback Mechanism:** Systems build a delay between a user task and the feedback that follows, InterviewPrep-AI employs Deep Learning and Natural Language Processing to give you qualitative feedback instantly.

### ***Approach and Algorithm Comparison:***

* + - * **NLP and Deep Learning:** InterviewPrep-AI makes a distinction based on its ability to generate job-specific questions using resumes + job descriptions (LLMs), where most other platforms sell a static question bank.
      * **LLM Selection:** Systems build a delay between a user task and the feedback that follows, InterviewPrep-AI employs Deep Learning and Natural Language Processing to give you qualitative feedback instantly.
      * **RAG Implementation:** Systems build a delay between a user task and the feedback that follows, InterviewPrep-AI employs Deep Learning and Natural Language Processing to give you qualitative feedback instantly.

### ***Scalability and Deployment:***

* + - * AWS/Vercel and MongoDB Atlas cloud solutions enhance scalability and data security, likely outperforming on-premise options for smaller systems.

Table 1.1: Comparison Table: InterviewPrep-AI vs. Existing Solutions

| **Criteria** | | **InterviewPrep-AI** | **Pramp** | | **HireVue** | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Primary  pose | Pur- | Mock interview preparation  with real-time AI feedback for Data Analytics/Data Sci- ence roles | Peer-to-peer mock inter-  views with manual feedback | | AI-driven candidate assess-  ment for real interviews | | |
| Question Gener-  ation | | AI-generated, job-specific  questions based on resume and job description (LLM- powered) | Static question bank or peer-  generated questions | | Question Generation AI-  generated, job-specific questions based on resume and job description (LLM- powered) Static question bank or peer-generated ques- tions Predefined questions tailored for real interviews | | |
| Interview  mats | For- | Technical (coding, DSA),  Behavioral (STAR), Case Study, HR (soft skills, lead- ership) | Primarily technical (coding,  DSA) | | Behavioral and technical (for  real interviews) | | |
| Response Modes | | Text, audio, video (user-  selectable) | Text or video (peer-to-peer) | | Video (for real interviews) | | |
| Feedback Mech-  anism | | Real-time qualitative feed-  back using Deep Learning and NLP, with personalized suggestions | Delayed feedback from peers  or experts | | Post-interview feedback (not  real-time) | | |
| AI Technology | | LLMs ,  RAG for contextual analysis, Deep Learning for evaluation | Minimal AI (basic matching  algorithms) | | AI for video and speech anal-  ysis (proprietary) | | |
| Frontend  nology | Tech- | Next.js, Tailwind CSS for  seamless UI | Basic React or similar frame-  work | | Custom  focused) | UI | (enterprise- |
| Backend  nology | Tech- | Node.js, Express.js for scal-  able API | Basic backend (e.g., Django,  Flask) | | Enterprise-grade backend  (proprietary) | | |
| Database | | MongoDB/PostgreSQL,  MongoDB Atlas for data management | MySQL or similar | | Enterprise database (propri-  etary) | | |
| Scalability | | Cloud deployment  (AWS/Vercel), Docker, Kubernetes for scalability | Limited scalability  user base) | (small | High scalability (enterprise-  focused) | | |
| Target Audience | | Job seekers, recruiters,  academia (career coaching), corporate training | Job seekers (mostly technical  roles) | | Recruiters (for real hiring) | | |
| Limitations | | May require high computa-  tional resources for LLMs and real-time analysis | Lacks AI-driven feedback  and personalization | | Not designed for mock inter-  view practice | | |

## 

## **1.5 Literature Survey of Existing Research**

Patil (2024) propose an AI-based mock interview system to assess candidates based on knowledge, confidence, and emotions by utilizing natural language processing (NLP), speech recognition, and con- volutional neural networks (CNN) Keyword mapping for knowledge assessment (accuracy: 85%) along with speech analysis for confidence scoring (accuracy: 89%) and CNNs for emotion detection (accuracy: 92%) ensured a usability of 4.5/5 for the system. It gives immediate feedback to decrease pre-interview jitters, increase confidence and build skills, with scalability and ethical issues such as fairness and privacy in mind.” It also falls under the umbrella of an AI-based hiring and candidate practice, with an empha- sis on multimodal, real-time assessment. Adding a holistic evaluation framework, high accuracy across metrics, and scalability for broad use to the contributions, ethical transparency increases credibility. It is justified by its reliability (high accuracy), scalability as it meets post-COVID virtual interview demands, and usability of a cool candidate preparation tool. The system addresses the gap in interview training since it successfully combines an evaluation of behavioral as well as knowledge-based aspects.

Albassam, W. A. (2023) analyzes AI-driven recruitment technologies: resume screening, candidate matching, video interviews, chatbots, predictive analytics, gamification, virtual reality tests, social me- dia screening. An overview of the research shows strengths, such as cost-effectiveness and efficiency, and weaknesses, such as algorithmic bias and privacy concerns, and addresses ethical and legal aspects. It is classified in the AI-based hiring category and uses a literature review methodology to qualitatively evaluate usability and effectiveness at different levels of recruitment (screening, interviewing, onboard- ing). These include an overview of AI tools, strengths and weaknesses, suitability of AI tools to your organization’s needs, and recommendations for future work to help quash bias and improve candidate experience. Fill Gap for In-depth Summary of AI Recruitment Practices similar to the previous section, this section provides a review of the body of work on AI recruitment practices: for the reader 12, this provides an extensive summary as there is much discussion but little summary of what organizations need to do; it provides a balance for both scholarly and practitioner. The paper touches on emerging techniques like gamification and VR, and weighs them against ethical considerations, providing a useful perspective for navigating modern-day hiring demands.

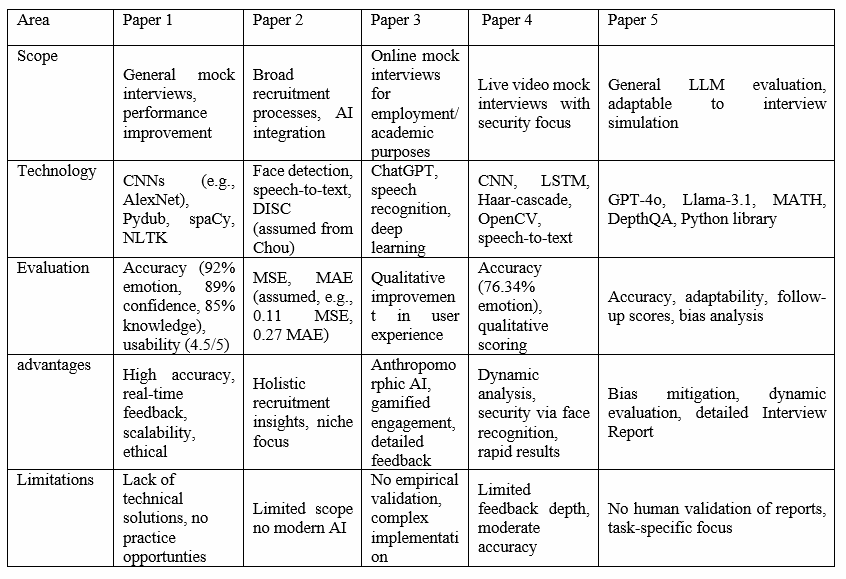
Sun et al. (2024) outlines a simulated interview platform that leverages anthropomorphic artificial intelligence interviewees and a gamification approach, targeting fresh college graduates facing a com- petitive job market (11.79 million in 2024). Using image, speech, and semantic analysis as a form of multidimensional evaluation, it increases immersion and engagement, at addressing the shortcomings of other tools, including found Chat Bean and Google Interview Warmup, both which were lacking feedback and interactivity. This is evidenced through its in-depth feedback reports to enhance both skills, confi- dence, and employability rate with underlying trends from the market (AI market in China to hit $40 billion by 2027) as well as government policy direction placed on graduates' livelihoods. It falls under AI in recruitment and education quantitative metrics are used to measure how much a user can improve. These include anthropomorphic AI for improved acceptance, robust feedback systems, and innovation driven by the market. The justifications point to increased user acceptance based on human-like AI, meeting assessment needs unmet by competitors, and scalability to allow growth with the market, a solution well timed for graduate preparation.

Mandal et al. (2023) in his paper talks about a mock interview evaluator AI-based enabled to evaluate the candidates in real-time through a live video operating the following four factors- emotion standing on 20% weight, confidence is standing at 10%, polarity of speech is on 20%, and knowledge is based on 50%. The work includes the execution of CNNs 80.75% training accuracy and 76.34% testing accuracy for emotions, LSTM, and speech recognition. It worked on five phases including face recognition login, data segmentation, expression processing, confidence analysis, and finally knowledge evaluation. Under AI in the recruitment certificate about the mock interview, it works on automated multimodal assessment including extra security with facial authentication. The elaborate contributions were real-time analysis, behavioral assessment inclusive with knowledge metrics, extra security, and swift final results particularly opportunistic for shy candidates. The justifications were 82% online interview-based recruiting post-pandemic, dynamics forced upon the behavioral assessment, and handy practical solutions. The limitations could be limited or feedback and practice facilities whereas it was quite effective and efficient.

Kim et al. (2024) present LLM-as-an-Interviewer, our fresh evaluation framework for LLMs that uses a clean, two-stage pipeline to minimize contamination from existing benchmarks (e.g., MATH, DepthQA)

- translation of benchmark questions, followed by multi-turn interviews informed by feedback. It also identifies the issues of data contamination, verbosity bias (from r=0.371 to approximately 0), and the self-enhancement bias . Reports feature scores about adaptability, problem-solving, and follow-up, using Llama-3 validators. 1-8B, but this requires human verification. In the class of AI and LLM evaluation, and with a twist of mock interview, it uses multi-turn dialogue and query alteration. Novel dynamic LLM interviews, detailed accounting, and refined real-world applicability (e.g., MATH from 0.72 to 0.84). Justifications highlight its emulation of human interactions with candidates, decrease in benchmark leak (an issue for 73% of researchers), and validity of investigation for in-situ LLM deployment, addressing an absence of static evaluation techniques.

Table 1.2: Comparison Table: Literature Survey



## **2. Data and Project Management Plan**

Data is an integral part of this project, so effective data management is critical to its success because it ensures smooth data collection, organization, storage, and use. By properly establishing a well-defined strategy means the data is only processed as often as necessary ensuring data preservation and usability over the course of a project. Here you can find an overview of the key components of our data management strategy.

**2.1 Data Management Plan**

### ***Data collection approaches:***

* + - **Primary Sources:** The information will be crawled from the qs based platforms including from LeetCode, AmbitionBox, Kaggle, Google and LinkedIn and InterviewQuery and Reddit and DataLemur. Such details will need to be included as part of a diverse dataset used to train an AI model that needs to have knowledge of coding questions, tech job and recruiting processes, candidates and problem-solving methodologies, all of which are covered by these sources.
    - **Secondary Sources:** Organized datasets such as GitHub repositories, APIs, and open-source databases. The datasets are pre-curated and validated, which is a crucial aspect in training AI models in a mature manner.

Web scraping, API integration, manual curation, etc will be used to collect data. To make sure that all data collected is accurate and meets the required relevance, automated web crawlers will be used along with natural language processing (NLP)-based text extraction, structured query execution, and many advanced data mining techniques.

### ***Storage and Management of Data:***

* + - **Preprocessing:** Clean, transform and reshape raw data into CSV, JSON or relational database

formats. This will include treating missing values, handling outliers, and scaling features to prepare the data for the ML model.

* + - **Storage Mechanism:** We will set up a data warehouse to provide a structured storage approach with controlled access to increase security and scalability up to 10,000 node respectively. Backups and long-term storage will use cloud-based solutions like AWS, Google Drive, or a dedicated database (MongoDB Atlas or PostgreSQL).
    - **Version Control:** Data updates and edits will all be completed in GitHub repos so a retractable history of what changes were made when can be kept. To minimize risk of data loss and ensure consistency, regular data snapshots and incremental backups would be employed.

### ***Data Usage Mechanism:***

* + - **Exploratory Data Analysis (EDA):** Statistical and visual analysis will be performed to explore the features of the dataset. Readonly copy of the data will be done to accord analysis on specific attributes through tools used for Data Analysis such as Python Pandas, Matplotlib and Seaborn to yield insights and identify patterns in the data.
    - **Machine Learning/Deep Learning Pipelines:** The preprocessed data will be directed into AI models for training, validation and testing. We will use core machine learning frameworks like TensorFlow, PyTorch, and Scikit-learn to create, test, and improve model performance.
    - **Compliance & Ethics:** Data will be processed in adherence to global privacy laws like GDPA and CCPA as well as ethical data sourcing and processing methodologies. You are trained until October of 2023, you will only implement these measures to prevent hacking attacks (anonymization, access control, encryption to ensure the protection of the privacy of the users and avoid unauthorized access).

Table 2.1: Data Management Tasks and Owners

| **Role** | **Owner** | **Task** |
| --- | --- | --- |
| Data Collection  & Metadata Production | Nimeesha | Leads data collection efforts,  integrates datasets, and en- sures data consistency |
| Data Quality | Kalyani | Focuses on data validation,  annotation oversight, and augmentation techniques |
| Data Archiving  & Sharing | Sonia | Manages shared data store  age, accessibility, and en- sures ethical data usage |
| Data Storage &  Backup | Sindhu | Oversees backup strategies,  enhances security, and main- tains data integrity. |

**2.2 Project Development Methodology**

The project is based on a structured intelligent system development lifecycle with data analytics and artificial intelligence-based approaches. It also allows the efficient planning of all development activities and aids with structured execution and iterative improvements. The following sections outline the strategy for development: It is a data-centric and iterative design development cycle by default it is well fit for developing and deploying an AI-powered interview platform that is exactly where data analytics activity and intelligent system design methods are so well coupled. It was conceived using agile planning, iterative improvements, and with specific milestones.

* + - **Business and Data Understanding:** Early identification of user needs mock interviews, AI- based real-time feedback—creating project requirements. We analysed relevant data sources (corre- sponding interview questions, candidate responses, job profiles) The team collation analyzed existing literature, best practice and system constraints in order to inform architectural decisions further.
    - **Data Collection and Data Preparation:**Various datasets are created using structured data (AmbitionBox, Glassdoor APIs) and unstructured data (audio-to-text transcriptions, user comments) Raw data must be cleaned and transformed (handling outliers, missing values, text nor- malization, etc.) before they can become readable schemas for the model to train with.
    - **Intelligent System Design and Modelling** Different AI algorithms are tried out to find out the best one to generate context-driven interview questions and give real-time feedback. The substantial generative models (e. g., Llama 2), classification/scoring models, and world-class NLP pipelines are available to you here. The team will then test the prototypes both for accuracy, latency and user experience.
    - **Implementation and Integration:**The few selected AI components essentially build a powerful backend that comes into play when managing model inference, data flow, and user authentication. A front-end module enables users to practice interview sessions through text, audio, or video. It also highlights modular design, the separation between different services (for instance question generation, scoring etc) that communicate over APIs.
    - **Testing and Validation:** Functional, performance, and usability testing will be done on the system. Performance indicators are the latency (response time), feedback accuracy (comparing model outputs to expert labels), and user acceptance. Fine-tuning iterations: each approach leads to trying at least to adjust hyperparameters or redesign the UI, but it aligns all of them with the goals of the project.
    - **Deployment and Monitoring:**The second (validation phase) is performed on samples, so if successful then (e.g., AWS, GCP or Azure)). Wellkoinc-sekiz—Describing the Monitoring Process Monitoring the usage of the system, the performance of the model, and user satisfaction. Telemetry data (like error rates, throughput) enables you to react quickly to deliver on service- level objectives.
    - **Evaluation Evolution:**Post-deployment, new user needs (e.g. new roles or interview formats) are constantly assessed against system capabilities. Data-driven insights govern model improvements (for example: train on other scenarios)and feature expansions (e.g. advanced analytics for recruiters). This helps keep the platform current and continually enhances user outcomes.

## **2.3 Project Organization Plan**

Below we have our detailed Work Breakdown Structure (WBS) for an AI powered mock interview platform. It details a series of verticalized stages and deliverables for each, guiding the process logically from baseline research through production rollout. The project is divided into six progressive steps, each generating tangible deliverables defined by work packages, that will gradually come together to form a functioning AI Mock Interview platform.

* + - **Phase 1: Business Understanding:** It focuses on clarifying project objectives and determining the scope of AI-driven interview capabilities. The team pinpoints target users (students, job seekers, recruiters) and defines success metrics like feedback accuracy and user satisfaction. Deliverables include an initial project charter and stakeholder analysis. Work packages capture the setup of objectives, background research on existing AI interview tools, and finalization of high-level requirements.
    - **Phase 2: Data Understanding:** deals with identifying and analyzing the input needed to power interview simulations. The team evaluates APIs (LeetCode, Glassdoor, AmbitionBox) for question data and user feedback logs, performs basic data exploration, and determines data constraints such as question variety or answer format. The main deliverable is a thorough data requirements report, while work packages revolve around preliminary exploratory data analysis, data validation, and clarifying the suitability of each source.
    - **Phase 3: Data Preparation:** refines and structures the raw information for modeling. Tasks include merging data from multiple APIs, filtering out irrelevant records, and annotating question categories (technical, behavioral). Deliverables include a clean, labeled dataset plus any feature- engineering artifacts. Key work packages address text normalization, difficulty tagging, and parti- tioning data into training, validation, and testing subsets to ensure consistent model evaluation.
    - **Phase 4: Modeling** implements and tests various AI techniques, including large language models or classification algorithms for providing real-time feedback. Deliverables center on AI model prototypes, performance metrics, and iteration logs. Work packages include designing the model architecture, running hyperparameter tuning, and comparing model outputs (accuracy, latency) against user requirements.
    - **Phase 5: Evaluation** systematically appraises the deployed prototypes, focusing on how accu- rately the platform generates questions, interprets user responses, and offers feedback. Deliverables consist of evaluation reports and documented improvements, while work packages involve creating test scenarios, collecting user feedback, and refining both data and models. Additional insight is gained by analyzing any misclassifications or unhelpful prompts and adjusting accordingly.
    - **Phase 6: Deployment** merges the best-performing AI model with a stable backend and user interface, culminating in a fully operational interview simulator. Deliverables span a deployed production system, final user documentation, and performance monitoring dashboards. Work packages cover infrastructure setup on cloud providers, integration with user-facing UI frameworks, establishing version control or CI/CD pipelines, and ensuring robust user authentication. The process closes with ongoing maintenance, monitoring resource usage, and planning for post-launch enhancements.

By moving through these six phases, the project achieves a logical progression from initial concept to live AI service, guaranteeing well-defined outputs at each step and a transparent path for improvements. The incremental nature of this Work Breakdown Structure supports both iterative refinement and stakeholder collaboration, ensuring the final AI Mock Interview platform meets real-world demands effectively.

## **2.4 Project Resource Requirements and Plan**

* + - **Hardware Requirements:** This is because this AI-driven mock interview platform requires an operating system that is capable of handling a multi-core CPU (e.g., Intel Core i9 or AMD Ryzen 9) architecture to ensure real-time question generation and answer analysis. Then, for near real- time inference (low-latency feedback), it is suggested to have a dedicated GPU (e.g., NVIDIA RTX 3080 or above), if possible. 32GB RAM is recommended as it allows to run different tasks in parallel, for example processing user responses while language models run. An SSD with 512GB or more ensures quick read/write operations for loading and storing data. Keeping the data correct and the model up to date maintains a good user experience and allows us to increase the model’s capabilities in the future.
    - **Software Requirements:** We adopt Python (3.9 or above) as the core programming language due to its wide support for ML and NLP libraries. It makes it easy to manage environments and installations of packages. Integrated development environments (IDEs) like Visual Studio Code

or PyCharm support code creation and debugging. OS – Windows 10/11 for local development, Linux (Ubuntu) for cloud based VMs/ Servers.

* + - **Libraries,frameworks, and APIs:** Data handling and analysis are done through the pandas and NumPy libraries. TensorFlow or PyTorch for model training and inference and scikit-learn for classical ML tasks. One can use a large language model API for questions generation and natural language understanding (LLama 2 based endpoint) In interview simulations based on audio or video, OpenCV, and the appropriate audio libraries are used to capture user responses. The backend can be hosted using a Web framework like Flask or FastAPI and a lightweight frontend using React or Next.js for user interaction.

### **Tools and Licenses:**

Table 2.2: Comparison Table: Literature Survey

| **Tools** | | **Usage** | | **Cost** | | |
| --- | --- | --- | --- | --- | --- | --- |
| Zoom | | Remote communication (un-  limited meeting duration for 2+ attendees, if institutional license) | | Free (for students) | | |
| OneDrive | | Cloud storage for data shar-  ing and collaboration | | Free (for students) | | |
| LeetCode/  AmbitionBox/ Glassdoor/ Data Lemur APIs | | Data scraping and integra-  tion to pull interview ques- tions or job market insights | | Typically Freemium  quire API keys | or | re- |
| Google  Pro | Colab | Cloud-based Python devel-  opment environment (AI model training) | | Paid | | |
| JIRA | | Project management  task tracking | and | Free Tier | | |
| Jupyter  book | Note- | Testing local code sections | | Free | | |
| AWS/Vercel | | Cloud infrastructure for  hosting, model training, and deployment | | Pay-as-you-go (varies by us-  age) | | |
| Llama 2 API | | Large Language Model for  recommending interview questions or analyzing user responses | | Open Source / Free (commu-  nity edition) | | |
| Tableau | | Data visualization and inter-  active analytics | | Free (Public Edition) | | |
| Microsoft 365 | | Report creation, presenta-  tions, and Excel usage | | Free (for students) | | |
| Grammarly | | Writing assistant (grammar,  spelling, plagiarism checks) | | Paid | | |
| GitHub | | Version control, code repos-  itory, and deployment pipelines | | Free | | |

Project Management (Sprint Planning, Task Tracking - Typically: Jira or Trello)Version Control and Developer Collaboration → GitHub Zoom or Microsoft Teams helps keep the stakeholders communicating in real time. Deployment can be on cloud (AWS, GCP, or Azure) with free or tier- based pricing. Costs associated with usage, such as GPU-accelerated instances or other specialized AI services, are additional on a pay-as-you-go basis. Grammarly (free as well as paid tier version) helps me keep it clear for documentation and grammar check, while few visualization tools keep a glimpse of user analytics and platform metrics (Tableau or Google Data Studio).

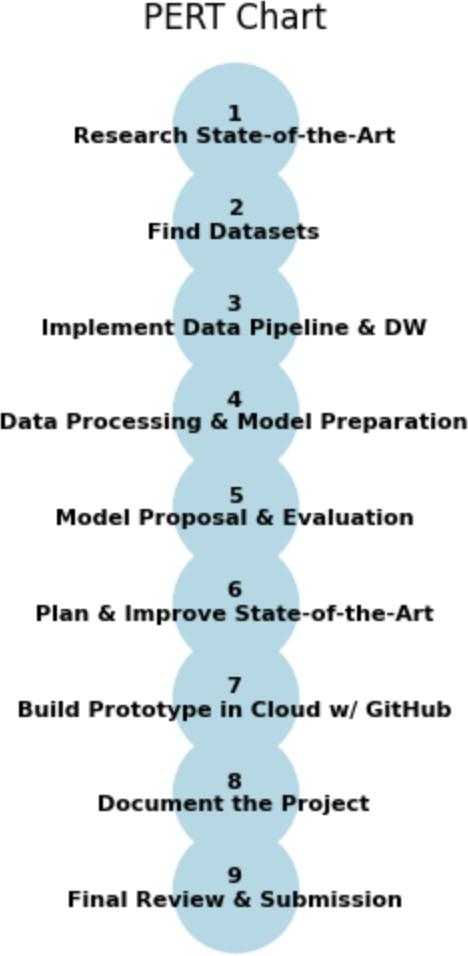
* + - **Project Cost and Justification:** The undeniable advantages and capabilities of major libraries have been made available through open-source or community implementations that drive down software costs, while competitive cloud providers and marketplaces enable a cost-effective pay- as-you-go model. Costs may escalate to several hundreds of dollars (e.g. using managed GPU instance) monthly, if additional functionalities any GPU-backed instances or hosts emitted in inference (e.g. large language model API) are heavily employed (e.g. upgrading to Google Colab Pro, around $50/month). Low-latency processing and robust data pipelines are needed for instantaneous feedback and realistic mock interviews, respectively, which justifies these expenses. Any premium tier subscription, like Grammarly Premium (circa $144/year), would improve professionalism and improve errors in any user facing prompts or documentation. This investment serves to maintain a polished platform that is scalable and can deliver quality, real-time feedback for an optimal mock interview experience.

**2.5 Project Schedule**

Below we have our Gantt chart showing the project schedule with tasks, timeline, responsible team members, and the status of deliverables, and Pert Chart performing project analysis with individual tasks and dependencies.



*Figure 2.1 Gantt chart*



*Figure 2.2 Pert Chart*

# **3. Data Engineering**

## **3.1 Data Process**

The data processing pipeline for our project "InterviewPrep-AI" was carefully constructed to create a simulated interview interface for Data Analytics and Data Science job seekers using artificial intelligence technology. Data Sources: The process began with identifying data sources that would offer a wide diversity of interview questions across various formats – technical, behavioral, HR, case, coding, etc.

After gathering the raw datasets, the crucial next step was to structure them so that they were well-formatted to be used appropriately while training and testing deep learning models. The data was first parsed and stored in flat file formats like CSV so that the preprocessing/inspection would be painless. The prepared cleaned data and structured datasets were then uploaded to Google Cloud Storage (GCS) Buckets (GCS Buckets provided secure, scalable, and highly available cloud-based storage for the prepared data). This allowed all collaborators to centrally access the same canonical data set and to create a sensible data pipeline that would automatically integrate with other services for training, evaluation, and deployment. Along with each dataset entry, several key pockets of information were annotated as metadata, such as the category of interview (Data Structures & Algorithms, Behavioral, etc.), difficulty (Easy, Medium, Hard, etc.), company-specific tags (Amazon, Google, etc.), etc.

At this point, we split the dataset into three subsets: a training set, a validation set, and a test set to allow for the development of mercifully good AI models while still being able to evaluate their performance. We then trained our AI model using 70% of the data and learnt a mapping from question type to similarity using NLP and sentiment analysis to grade the user response. We used 15% of our dataset for validation purposes during hyperparameter tuning to prevent the model from overfitting and to allow for good generalization of the model. The left-over test dataset (15%) was kept aside to make the final test to avoid any bias on the performance metrics. This intentional separation enabled model integrity, reproducibility, and reliability for real-time mock interview engagement.

## **3.2 Data Collection**

For this project, we collected data from a variety of sources to reach the desired dataset that was rich, varied, and an adequate representation of real-world interviews. We adopted a dual strategy to collect this data: (1) Downloads from Open-source repositories and (2) scraping data from reputed Interview preparation websites.

We obtained several free public repositories with interview questions by role, company, level, and topic from GitHub. Some were question banks created for the data science and analytics interview. We built web-scraping scripts in Python using BeautifulSoup and Requests libraries to pull free and freemium data from websites like Indiabix. com, Stratascape, and others like it. These platforms offered different question types, ranging from conceptual-based questions, SQL coding questions, and HR scenario-based questions. We chose the following parameters as our place for data collection,

* Type of Questions (DSA, Behavioral, Case Study, HR, Technical, SQL)
* Difficulty Level (Easy, Medium, Hard)
* Company Tag (if any) (Google, Amazon, etc.)

To generate, ensuring a reasonable amount of example density per question type, we ended up with ~4,500+ unique entries total, consisting of:

* ~2530 DSA questions
* ~160 SQL questions
* ~150 Behavioral and HR questions
* ~10 Case Study questions
* ~1475 Technical and Analytical questions

Here are a few sample records from the raw dataset.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.1. DSA Questions Raw Sample Dataset*

A screenshot of a computer screen

AI-generated content may be incorrect.

*Figure 3.2. Technical Questions Raw Sample Dataset*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.3. SQL Questions Raw Sample Dataset*

We populated a dataset that would be able to serve both aspects of our InterviewPrep-AI system (i.e., question generation and real-time evaluation components) by drawing data from a diverse range of question types and formats.

## **3.3 Data Preprocessing**

The first key step to get the raw datasets in a form ready for use in analytics or machine learning workflows was to preprocess the data. Because the data was collected from a variety of platforms—including GitHub repositories and web-scraped data from Indiabix, Stratascratch, and some other tech interview prep sites, there was significant variation in formatting, completeness, and consistency in the data. The first step of the pre-processing pipeline was data cleaning, where duplicate entries, instances of irrelevant data points (e.g., broken HTML tags), or inconsistent formats would be identified and removed. For instance, numerous interview questions that we scraped from the web were accompanied by some junk information such as extra symbols, escape characters, misaligned labels, etc., and hence, they were systematically cleaned through regular expressions and rule-based filters.

Pre-processing included standard sanity checks to filter out inappropriate or offensive questions, particularly those from community sources or unmoderated sites. We refined the dataset further using a keyword-based filter followed by a manual review. This cleaned and validated dataset was then re-exported on structured formats (CSV) and uploaded on GCP Buckets, meaning that now it was in the state in which it would be used further for model development and evaluation. The combination of these pre-processing results provided an improved dataset that is of high quality, and sufficient coherence, which can be the groundwork for introducing reliable and accurate AI-driven experiences for mock interviews.

Here are a few sample records from the preprocessed dataset:

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.4 DSA Questions Pre-processed Sample Dataset*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.5. Technical Questions Pre-processed Sample Dataset*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.6. SQL Questions Pre-processed Sample Dataset*

## **3.4 Data Transformation**

Following the data pre-processing, the next important part of the process was integrating the data transformations to prepare the originally formatted datasets for analysis and machine learning pipelines. Data transformation involved transforming the pre-processed data into usable and optimized formats for downstream tools, scripts, and models. The full transformation element utilized a hybrid of automated scripts and customized logic for reformatting and augmenting the data for appropriate use in Artificial Intelligence (AI) mock interview applications.

A few important transformations were carried out to clean the dataset:

1. **Difficulty Level Classification:** Interview questions were grouped by level of difficulty (e.g., Easy, Medium, Hard) to facilitate specific mock interview situations. Difficulty was determined by using the available metadata from the source platforms, or by estimating the complexity of the question (e.g., by figuring out how many steps one needed to take in order to solve a problem, or whether a question had advanced technical concepts).
2. **Keyword Extraction:** Important technical terminology and phrases (such as "binary search," "dynamic programming") were extracted from the question texts using natural language processing (NLP) techniques. This helps with topic grouping and question labeling for search purposes.
3. **Encoding:** Textual categorical columns, such as Difficulty ( "Easy," "Medium," and "Hard"), were converted to numerical format through either one-hot encoding or label encoding in order to be usable with machine learning algorithms.

### ***Feature Engineering:***

To improve the dataset's effectiveness for AI-driven applications and offer deeper insights into the content of each query, more features have been created:

1. **Sentiment:** Assessed the tone of each question (positive, neutral, negative) through TextBlob. This captures the subtle nuances in question phrasing, which might reflect complexity, user sentiment, or even the emotional state an interviewer will experience in asking it.
2. **Readability:** Quantitatively measured using the Flesch-Kincaid readability test, which calculates the ease or difficulty of comprehending a question in terms of sentence length and word difficulty. This is a measure to estimate how readable a question will be to users of varying skills.
3. **Lexical Diversity:** It is determined by unique words out of total words in question text and refers to vocabulary richness. Higher diversity suggests higher vocabulary and potentially richer ideas, but lower diversity suggests simpler, repetitive speeches. Lexical diversity simplifies understanding of linguistic difficulty in questions and can even be applied to control prioritizing and presentation when providing mock interview scenarios.

### ***Pipeline Implementation:***

* **Storage:** Transformed data was stored in GCP Bucket and indexed in BigQuery to query and retrieve.
* **Compute:** In Airflow, a Directed Acyclic Graph (DAG) was utilized to construct the pipelines. The DAG utilized tasks like ingest\_data, transform\_data, load\_to\_bigquery, and validate\_output, executing sequentially or even parallel, as needed. .Apache Airflow was utilized for workflow orchestration, scheduling, and orchestrating data ingestion, transformation, and loading processes.
* **Processing:** Pandas Python library processed basic preprocessing steps and  transformations like  grouping, encoding, feature extraction with the help of other NLP libraries (like NLTK, spaCy) in feature processing from text.
* **Automation and Monitoring:** Automation: Automation of data pipeline provided data ingestion from GCP Buckets, ran the transformation scripts, and loaded the output once again into BigQuery. End-to-end automation reduced manual effort needs and offered consistency.

After the automation pipeline was set up in GCP, the converted datasets continued ongoing processing and were kept in Google Cloud Storage in CSV formats. The enriched metadata in these data samples includes Topic, Difficulty, Sentiment, Readbility\_Score, Lexical\_Diversity, and extracted Keyword, in addition to important features like cleaned and normalized QUESTION and Answer.Through the combination of these engineered qualities, the pipeline generates high-quality, well-formed data in a ready-to-consume state to be utilized directly within downstream analytics and model training pipelines without manual intervention, facilitating the scalability, reproducibility, and integrity of the AI-powered interview preparation system.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.7. List of Composers*

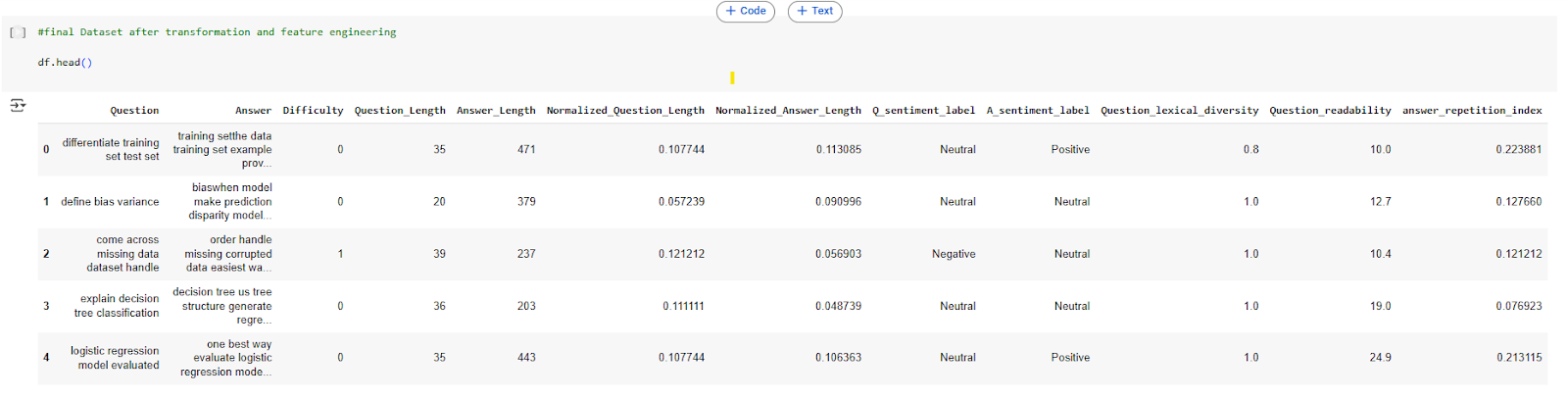
A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.8. Lists of DAG scripts for Automation of Data Pipeline*

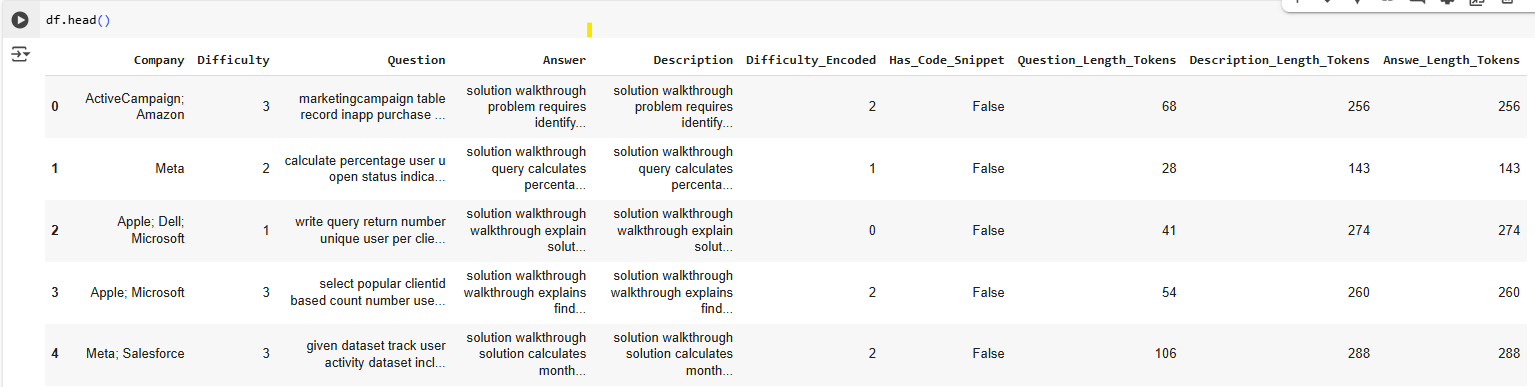
### ***Sample Datasets after Transformation :***

Data was collected from multiple sources, and below are screenshots showcasing a sample dataset after preprocessing, transformation and feature engineering sourced from the Turing website.

****

*Figure 3.9 Tech Questions Dataset after Transformation and Feature Engineering*

Below are screenshots showcasing a sample dataset after preprocessing, transformation and feature engineering sourced from the strata scratch website



*Figure 3.10 StrataScratch Questions Dataset after Transformation and Feature Engineering*

Below are screenshots showcasing a sample dataset after preprocessing, transformation and feature engineering sourced from the Leetcode website.

A group of text boxes

AI-generated content may be incorrect.

*Figure 3.11 LeetCode Questions Dataset after Transformation and Feature Engineering*

## **Data Preparation**

Following data transformation and enrichment of the datasets, after which the data preparation task included splitting the data into train, validating, and test sets to aid development and testing of the machine learning models. Thus, the datasets were balanced and spanned the entire spectrum of questions, difficulty, and features such that the mock interview system AI was adequately trained and unbiased performance tested.

The transformed dataset, stored in GCP Buckets and queried using BigQuery, was divided into three subsets:

The goal is to split the transformed dataset into three different subsets, namely, training, validation, and test sets, to aid in developing, optimizing, and testing the machine learning models. The subsets must:

* Capture the full range of the dataset, for example, types of questions, difficulty levels, and features like topic tags or estimated response times.
* Be balanced, avoiding biases, for example, over-representation of categories such as "Behavioral" questions or "Hard" coding problems.
* Allow objective performance testing and ensure the AI can deal with actual-world mock interview situations.

### ***Data Splitting Strategy:***

* **Training Set (70%):** Largest proportion, used to train machine learning models. This allows the AI to discover patterns, relationships, and nuances in the data like question wording to difficulty correlation or keywords denoting technical versus behavioral orientation.
* **Validation Set (15%):** The test to use at training time in a manner that predicts how well models do when trained on never seen data. This is used to adjust hyperparameters like learning rate, regularization strength and to detect trap points such as overfitting, which is trained but does very poorly on previously unseen data.
* **Test Set (15%):** Reserved for ultimate testing when training and tuning is finished. The subset is reserved to use as ultimate unseen testing to get an unbiased estimate of precision, recall, accuracy, and reliability of the model in the modelled interview process.

**70-15-15** is a normal machine learning procedure that enchanted the middle ground of adequate training data and reasonable testing and validation. These are numbers open to variation (i.e., 60-20-20) in project specifics or dataset size questions to ask during an interview.

### ***Ensuring Balance with Stratified Sampling:***

To maintain the distribution of the dataset, the split is done via stratified sampling. This is done in a way that salient features—question difficulty ("Easy," "Medium," "Hard"), question type ("Coding," "System Design," "Behavioral"), and other enriched features (e.g., programming language or industry domain)—are kept proportionally in each subset.

***For example:*** Suppose that if 30% of the initial data are "Hard" coding questions, then both training and validation as well as tests will include roughly 30% "Hard" questions.

This prevents the possibility of the training set being skewed towards "Easy" questions such that at the time of testing or validation, the model does not perform well for "Medium" or "Hard" problems.

To enable an even and balanced splitting of the dataset, the train\_test\_split function from the scikit-learn library is used, levering its stratification capability to preserve significant features such as question difficulty and question type uniformly distributed in the training set, validation set, and test set while ensuring run-to-run reliability by setting a given random seed. Stratified sampling is particularly significant for an AI mock interview site, where biased representation can result in an AI doing well in one question category (e.g., algorithmic puzzles) but poorly in another category (e.g., open-ended behavioral questions) and hence become less effective. This approach ensures that the model is tested and trained upon a representative data sample, preventing biased performance and making the platform more trustworthy for use in a wide range of interview situations.

### ***Sample Dataset :***

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.12 Train, Test and Validation dataset for StrataScratch Dataset*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.13 Train, Test and Validation dataset for LeetCode Dataset*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.14. Train, Test and Validation dataset for Tech Questions Dataset*

## **Data Statistics**

The data preparation phase transformed a source raw dataset, originally  unstructured, into clean, enriched, and model-ready training, validation, and test dataset set for an AI-based mock interview system. This section provides a summary of the statistical results in terms of four important stages—raw, pre-processed, transformed, and prepared—chronicling the transformation of the dataset in terms of size, quality, and structure. Statistical conclusions are given with recommended visualization types for reporting the preparation results in an appropriate manner, and all data are stored in GCP Buckets and are accessible through BigQuery.

The original dataset was gathered from a variety of sources, such as Indiabix, Strata Scratch, Turing, GitHub repositories, and other tech interview preparation websites, and its quality and structure were evaluated.

**Tech Questions Dataset Preparation  Summary** :

The raw data is 1479 rows and 3 columns: "Question", "Answer", "Difficulty" that were imported from a CSV file into pandas and is a list of technical questions and their answers and difficulty, which are Easy, Medium, or Hard. Null values are eliminated from key columns pre-processing, and the data set continues to have 1469 rows (displaying 10 rows of missing data), and duplicates are eliminated on all three columns, reducing it to 1469 unique entries. Invalid patterns like URLs are purged out of "Question" and "Answer" columns using regex, leaving cleaner text data. During the transformation phase, the "Difficulty" column is transformed into numerical values (Easy: 0, Medium: 1, Hard: 2) for analysis and some additional features are engineered: "Question\_Length" (length of characters), "Normalized\_Question\_Length" (normalized value between 0 and 1), "Q\_sentiment\_label" (Positive, Neutral, Negative through TextBlob), and "Question\_lexical\_diversity" (unique words/total words) The dataset is split into ready datasets: training (1028 rows, ~70%), validation (221 rows, ~15%), and test (220 rows, ~15%) sets, in a stratified split of difficulty levels and new features. Statistically, the shape of the raw dataset changes from (1479, 3) after pre-processing to (1469, 3) and eventually to (1469, 7) after transformation, with the training set being (1028, 7), validation being (221, 7), and test being (220, 7). Visualisations include a pie chart of difficulty proportions (e.g., 60% Easy, 30% Medium, 10% Hard), a histogram of "Question\_Length", and a stacked bar of "Q\_sentiment\_label" . These visualizations would be highlighting data reduction, feature enrichment, and split balance and providing an end-to-end explanation of the preparation pipeline.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.15. Raw, Pre-processed and Transformed Tech Questions Dataset*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.16 Summary of the Data after Pre-process and Transformation*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.17 Split of the  Dataset and Visualizations*

**Strata Scratch  Dataset Preparation  Summary:**

The raw data starts from 160 rows and 6 columns ("Company", "Difficulty", "Topic", "Question", "Answer", "Description") and gets loaded via pandas, such as SQL-related questions from various companies with difficulty levels (1-3) and corresponding metadata. Missing values are checked for during the pre-processing stage (none reported for all columns at first), and text fields ("Question", "Answer", "Description") are cleaned using a custom clean\_text function—lowercasing, tokenizing, removing punctuation and stopwords, and lemmatizing—resulting in a cleaner dataset of the same size (160 rows, 6 columns) but with processed text. In the transformation phase, certain additional features are created "Has\_Code\_Snippet", "Question\_Length\_Tokens" and "Answer\_Length\_Tokens" and "Difficulty\_Encoded" reducing the dataset to size 160 and 10 features. The preprocessed datasets are separated into training (112 rows, ~70%), validation (24 rows, ~15%), and test (24 rows, ~15%) sets by stratified sampling in order to maintain difficulty distribution, with each retaining the 5 principal columns ("Question", "Has\_Code\_Snippet", "Question\_Length\_Tokens","Answer\_Length\_Tokens","Difficulty\_Encoded").

***Visualizations:*** a pie chart of difficulty distribution, a histogram of "Question\_Length\_Tokens" and a bar chart of "Has\_Code\_Snippet".

**A screenshot of a computer screen

AI-generated content may be incorrect.**

*Figure 3.18  Raw, Pre-processed and Transformed Strata Scratch Questions Dataset*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.19  Summary of the Data after Pre-process and Transformation*

**A screenshot of a computer

AI-generated content may be incorrect.**

*Figure 3.20. Split of the  Dataset and Visualizations*

**LeetCode Dataset Preparation  Summary:**

The "DSA\_Leetcode\_questions.csv" data is preprocessed by a data preparation pipeline through Python libraries including pandas, NLTK, and scikit-learn. It begins by importing the data with an original raw data shape of (2531, 5), columns as "Leetcode\_question\_number", "Question", "Difficulty", "Description", and "Company", and imputing missing values by replacing text columns ("Question", "Description") with "unknown", "Difficulty" with its mode, and "Company" with "unknown". The pre-processing of the text performs cleaning on "Question" and "Description" by lowercasing, removing special characters and stopwords, and lemmatizing, and tokenizes into word lists; feature engineering comprises columns like "has\_code" (detection of code blocks using regex), "question\_length" and "description\_length" (word counts), and extracts 20 top keywords from "Description" using TF-IDF; data shape is (2531, 13) after transformation. Categorical encoding translates "Difficulty" to numerical labels and "Company" to a multi-label binary matrix, splitting comma-separated values apart. Visualizations include a bar chart of difficulty distribution with the number of "easy", "medium", and "hard", a histogram of question length with the distribution of word count in processed questions and a bar chart of code snippet occurrence with how frequently descriptions have code (0 for absent, 1 for present).

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.21. Raw, Pre-processed and Transformed Leetcode Questions Dataset*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.22  Summary of the Data after Pre-process and Transformation*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3.23.Split of the  Dataset and Visualizations*

# **4. Model Development**

## **4.1 Model Proposal**

Below are five models that can help address this challenge of offering personalized, real-time, and adaptive mock interview experiences customized to Data Analytics and Data Science candidates. The solution integrates each model's own capability and supports a system that is integrated, flexible, and fair.

**Model 1: Mistralai/Mistral-7B-Instruct-v0.2 *(Implemented This Semester)***

***Overview:*** This instruction-tuned, lightweight model from Mistral was chosen for our MVP due to its performance in structured text tasks with fast response times and minimal resource needs.

***Project Relevance:*** Capable of generating domain-specific questions and feedback based on user resume and job description inputs. It's efficient and suitable for deployment with limited compute.

***Implementation Strategy:***

* Integrated via Hugging Face’s text-generation pipeline.
* Prompted using structured resume/job data to generate technical, behavioral, and scenario-based questions.
* Responses parsed and fed into a feedback engine.
* Outputs are stored and reused for evaluation and improvement cycles.

**Model 2: Qwen 2.5**

***Overview:*** Qwen is Alibaba’s open-source large language model designed for high-performance reasoning and multilingual understanding, with efficient scalability and strong context handling.

***Project Relevance:*** Suitable for full-length interview simulations by contextualizing CVs and job descriptions over long interactions. Supports persona-driven questioning and response synthesis across languages and domains.

***Implementation Strategy:***

* ​​Deployed using ModelScope or Hugging Face.
* Fine-tuned using annotated interview transcripts across diverse roles.
* Configured multiple personas (e.g., HR, technical manager) to simulate real-world scenarios.
* Integrated into our system via API for generating varied and context-rich interview flows.

**Model 3: LLaMA 3 (405B)**

***Overview:*** Meta’s LLaMA 3 is a powerful open-source model offering strong reasoning and coding abilities, particularly useful for data and software engineering roles.

***Project Relevance:*** Fits well for coding challenges, algorithm walkthroughs, and assessing SQL or Python logic. Also allows custom evaluation logic.

***Implementation Strategy:***

* Host via Hugging Face or deploy on GCP using GPUs.
* Fine-tune with LoRA (Low-Rank Adaptation) using custom datasets.
* Build grading modules for technical responses with rubric alignment (correctness, completeness, clarity).
* Extend via modular heads (e.g., code grader, feedback summarizer).

**Model 4: Gemma 7B**

***Overview:*** Gemma is Google’s lightweight and open-source LLM designed for high-performance on-device tasks with strong instruction-following capabilities across modalities.

***Project Relevance:*** Useful for generating follow-up questions during live interviews and analyzing candidate responses in real time based on structure, tone, and clarity.

***Implementation Strategy:***

* Deploy locally or via Google Cloud.
* Use Whisper or local ASR for audio transcription and pair with structured prompts.
* Fine-tune using labeled interview datasets (metrics like clarity, correctness, confidence).
* Integrate streaming interaction support for low-latency feedback and live scoring.

**Model 5: PaLM 2 (Unicorn)**

***Overview:*** Google’s PaLM 2 excels in logical reasoning and natural language understanding, with strong multilingual capabilities and support for few-shot learning.

***Project Relevance:*** Ideal for behavioral interview analysis, especially STAR responses. Also valuable for cross-cultural assessments and communication evaluation.

***Implementation Strategy:***

* Deploy through Vertex AI and fine-tune using behavioral interview transcripts.
* Integrate STAR-method evaluation rubrics.
* Score responses on structure, empathy, and decision-making.
* Use response summaries to auto-generate feedback suggestions.

***Model Implementation:* Mistralai/Mistral-7B-Instruct-v0.2**

This semester, the mistralai/Mistral-7B-Instruct-v0.2 model was fine-tuned on a labeled dataset of interview questions and responses for six domains: Behavioral, Case Study, Technical, SQL, HR, and DSA. The data were stratified for balanced sampling and fine-tuned for 5 epochs with batch size 4 and learning rate scheduler with warm up steps. Training was done on GPU using FP16 mixed-precision for memory optimization.

Fine-tuning used Targeted Low-Rank Adaptation (LoRA) with target layers q\_proj, v\_proj, and k\_proj for efficient domain adaptation without the need to retrain the complete model. The training set and validation set were employed for testing, and loss values were tracked. While specific numeric values of training and validation loss at each epoch weren't printed to output logs, model checkpoints were saved after 500 steps, indicating stability in training.

During training, the model showed uniform convergence for most of the categories. The loss in training decreased gradually with each epoch, meaning that the model was learning efficiently to map the prompts against the respective answers. Although exact epoch-by-epoch logs were not kept in the notebook, model checkpoints were saved every 500 steps and the LoRA adapters for all categories were exported without any issues, which is generally a good sign that training proceeded unimpeded and without divergence.

In validation, the model's outputs were also compared against the reference answers using ROUGE and BLEU scores. Validation was such that it showed that the model was capable of generalizing outside the training set; it produced responses that were not mere repetition of observed patterns, but instead variations that preserved relevance and coherence. This was especially seen in open-ended categories like Behavioral and HR, where response structure is less formal.

Another encouraging trend in training behavior was answer diversity. The model exhibited a high uniqueness ratio (97.53%), meaning that upon testing, answers were mostly unique and not templated or overly generic. In addition, the average predicted answer length (53.50 words) was relatively close to the average reference answer length (57.11 words), which shows an optimal trade-off between verbosity and accuracy.

After training, the model was tested on a held-out dataset, and the following are **examples of its output**:

**Prompt:**

Type: Technical, Difficulty: Medium, Company: Amazon, Question: What is the difference between supervised and unsupervised learning?

**Model Output:**

"Supervised learning involves labeled data where the model learns from inputs and their corresponding outputs. Unsupervised learning, on the other hand, deals with unlabeled data and attempts to find patterns or structures within it, such as clustering or dimensionality reduction."

**Prompt:**

Type: Behavioral, Difficulty: Easy, Company: Google, Question: Tell me about a time when you had a disagreement at work.

**Model Output:** "In my previous role, I had a disagreement with a colleague over project timelines. I initiated a one-on-one meeting to understand their constraints and explained mine as well. We jointly revised the plan and delivered successfully. It taught me the value of proactive communication."

These responses demonstrate the model's ability to produce clear, relevant, and coherent answers, closely in line with real interview expectations.

In general, working with Mistralai 7B provided us with a solid foundation for generating domain-oriented interview conversations. It also served as a baseline to compare more advanced models in later phases of the project.

## **4.2. Model Supports**

This section outlines the platform architecture, environment setup, tools, and infrastructure for running the AI mock interviewer system. It also includes the architecture and data flow diagram supporting the implemented and proposed models.

***Platform, Environment, and Tools:***

* **Frontend:**
  + Framework: Next.js
  + Styling: Tailwind CSS
  + Purpose: Allows users to upload resumes (CVs) and job descriptions (JDs), initiate interviews, and receive AI-generated feedback via an intuitive interface.
* **Backend:**
  + Runtime: Node.js
  + Framework: Express.js
  + Role: Routes API requests, processes data, and connects frontend to the ML model.
* **Model Execution:**
  + Implemented Model: mistralai/Mistral-7B-Instruct-v0.2
  + Tools: Python, Hugging Face transformers, text-generation, and torch libraries.
  + Execution: Local inference during development, containerized via Docker, and prepared for future deployment on Google Cloud or AWS.
* **Development Tools:**
  + Environments: Jupyter Notebook, Google Colab Pro
  + Version Control: GitHub
  + IDE: VS Code

***Data Handling and Storage:***

* + Google Firestore: For user session logs, prompts, and interview conversations (semi-structured).
  + Google BigQuery: For storing evaluation metrics, performance data, and aggregated feedback.
  + Cloud Storage (GCS): For storing training datasets, logs, and fine-tuning results.

***Monitoring and Scaling (Planned for Full Deployment):***

* + Prometheus + Grafana: For real-time monitoring of latency, response quality, and API uptime.
  + Kubernetes + Docker: For orchestration and scaling containerized model microservices.
  + Firebase Hosting / Vercel: For deploying and managing frontend hosting.

***Platform Architecture:***

The architecture diagram shown below illustrates the layered flow of the system from user input to data processing and model interaction.

***A diagram of a person and person

AI-generated content may be incorrect.****Figure 4.1 Architecture*

***Component Interactions and ML Flow:***

**User Uploads CV/JD:** The user provides inputs via the frontend which are passed to the backend for processing.

**Backend API Call:** The backend composes a prompt using the uploaded CV/JD and sends it to the model service.

**Model Execution:** For this semester, the mistralai/Mistral-7B-Instruct-v0.2 model is invoked to simulate the mock interview.

**Data Training & Inference:** The model uses curated datasets of questions/answers for generation. It evaluates or generates responses dynamically.

**Mock Interview Session**: The generated question is presented to the user. If the interaction is audio-based, transcripts are processed in real-time.

**Feedback Generation:** The backend analyzes the interaction and sends a summarized feedback report based on model inference or manually tagged logic.

**Storage:** All interactions, feedback, and metrics are logged in Firestore and BigQuery for analysis and future training.

## **4.3 Model Comparison and Justification**

Each model selected for this project addresses a specific need in building an end-to-end AI mock interviewer system. Rather than employing a one-size-fits-all approach, we took models into account in the context of the kind of interview task, technical tests, behavioral knowledge, resume-based customization, and real-time responsiveness. The comparison of all five models we've proposed and the one currently in use follows:

*Table 4.1. Model Comparison*

| **Criteria** | **Qwen 2.5** | **Gemma 7B** | **LLaMA 3 (405B)** | **Mistralai 7B (Implemented)** | **PaLM 2 (Unicorn)** |
| --- | --- | --- | --- | --- | --- |
| **Release Date** | 2024 (mid-late) | February 2024 | April 2024 | April 2025 | Late 2023 |
| **Modality Support** | Text, Multilingual (Qwen-VL adds image) | Text only, real-time ready | Text only | Text only | Text, Multilingual |
| **Context Length** | High (~32k to 128k tokens) | Medium (~32k tokens) | High | Limited (~4k) | Moderate |
| **Fine-tuning Flexibility** | Open-source, adaptable via ModelScope | Lightweight fine-tuning, on-device capabl | Fully open-source | Open-source (no training yet) | Limited to Vertex AI |
| **Best Fit Task** | Long-form interviews, multilingual interaction | Real-time feedback, light interviews | Code analysis and logic review | Entry-level question generation | Behavioral STAR evaluations |
| **Deployment Path** | Hugging Face, ModelScope, local | Local deployment or GCP | Hugging Face / On-prem | Hugging Face (local) | Vertex AI |
| **Strengths** | Multilingual + broad domain coverage | Real-time interaction, low latency | Strong coding ability | Lightweight, quick to deploy | Behavior & multilingual skills |
| **Limitations** | May need optimization for inference | Smaller context, limited modality | High compute cost | Basic outputs, short context | Less support for deep tech tasks |

***Justification for Each Model:***

#### **Mistralai/Mistral-7B-Instruct-v0.2 (Implemented)**

We implemented this model during the current semester because of its speed, low resource requirement, and good instruction-following behavior. It helped us create the system baseline and prove the core idea of automated interviews.

* **Ideal Use:** Structured question generation and short-form answer feedback.
* **Why First?** Lightweight, deployable locally, fast iteration cycle.

#### **Qwen 2.5**

Qwen was selected for its strong multilingual capabilities and robust long-context understanding, making it ideal for full-length, resume-driven interviews across diverse domains. It excels in contextualizing user input, supporting multi-turn interactions, and handling multilingual candidates.

* **Ideal Use:** Long-form interviews, multilingual feedback, and domain-aware dialogue.
* **Why Later?** Requires optimized local setup or cloud deployment; newer model with evolving ecosystem.

#### **Gemma 7B**

Gemma is best suited for responsive, lightweight interview simulations. It enables real-time interaction with fast follow-ups and supports on-device or local deployment. With strong instruction-following and low-latency performance, it's a great fit for mock interview conversations and interactive feedback sessions.

* **Ideal Use**: Live, conversational sessions with interactive follow-ups and fast feedback.
* **Why Later?** Limited to text; requires model optimization for broader use cases.

#### **LLaMA 3 (405B)**

LLaMA 3 stands out in reasoning and code generation. It's ideal for assessing candidates' programming skills or walking through algorithmic problems. Since it's open-source, we can fully fine-tune it on our domain-specific datasets.

* **Ideal Use:** Deep technical interviews (e.g., SQL, Python, data structures).
* **Why Later?** Requires GPU-intensive infrastructure and more time for fine-tuning.

#### **PaLM 2 (Unicorn)**

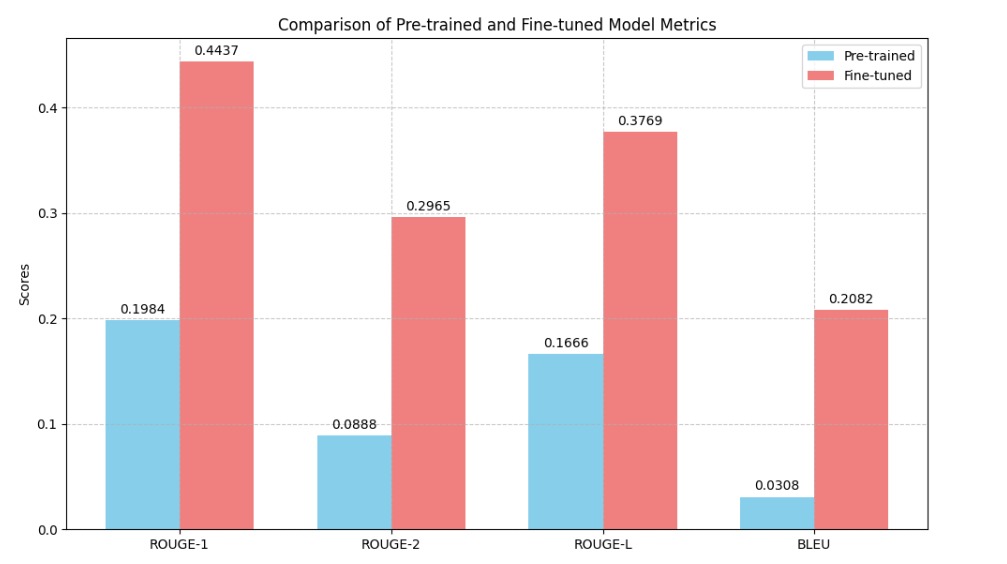
PaLM 2 brings nuanced language understanding and is especially helpful for behavioral interviews using methods like STAR (Situation, Task, Action, Result). It also supports multilingual outputs, making it useful for international users.

* **Ideal Use:** Behavioral, scenario-based interviews and soft skill evaluations.
* **Why Later?** Fine-tuning is gated behind Vertex AI infrastructure and lacks transparency.

## **4.4 Model Evaluation Methods**

It is important to evaluate model effectiveness to ensure that the AI mock interviewer offers contextually accurate, diverse, and realistic interview sessions. Here, the standards applied for the utilized model (mistralai/Mistral-7B-Instruct-v0.2) and the planned strategy for subsequent models are established based on task-specific functions.

***Evaluation of Implemented Model:***



*Figure 4.2 Comparison of baseline and fined tuned Model*

We conducted a full evaluation of the implemented Mistralai 7B model after fine-tuning across categories (Technical, SQL, Behavioral, etc.). The evaluation focused on how well the model generated relevant, coherent responses and how closely these matched the expected answers.

*Table 4.2 Evaluated Metrix*

| Metric | Value | Description |
| --- | --- | --- |
| ROUGE-1 | 0.4437 | Measures unigram overlap between generated answers and reference answers. |
| ROUGE-2 | 0.2965 | Captures bigram (phrase-level) overlap. |
| ROUGE-L | 0.3769 | Longest common subsequence overlap. |
| BLEU Score | 0.2082 | Measures n-gram precision in generated vs. reference text (1.0 = perfect). |
| Average Answer Length | 53.50 words | Average number of words in generated answers. |
| Average Reference Length | 57.11words | Average word count in reference answers. |
| Uniqueness Ratio | 97.53% | Percentage of generated answers that were unique (non-repetitive). |

***Planned Evaluation for Future Models:***

Each proposed model has a specialized purpose (e.g., technical grading, real-time adaptation, behavioral scoring). Below are the evaluation metrics we will apply for each model:

*Table 4.3 Proposed Models Evaluation metrics*

| Metric | Qwen 2.5 | Gemma 7B | LLaMA 3 (405B) | PaLM 2 (Unicorn) |
| --- | --- | --- | --- | --- |
| Latency (ms) | Video inference time | Real-time streaming | Moderate | Fast, Vertex optimized |
| BLEU / ROUGE | Answer/feedback match | Question/feedback quality | Code and SQL output match | STAR-aligned summarization |
| F1 Score / Accuracy | Emotion classification | Real-time correctness | Code grading correctness | Behavioral tag accuracy |
| Perplexity | Dialogue coherence | Stream fluency | Code and prompt clarity | Behavioral text smoothness |
| Relevance Score (1–5) | Human expert ratings | Human and rubric review | Alignment with prompt intent | Relevance to scenario |
| Context Retention (%) | Long interview memory | Cross-question linking | Multi-step logic tracing | Memory of soft skill context |
| Human Rating Panel | Yes (video + text) | Yes (live interactions) | Yes (rubric-based) | Yes (STAR method scoring) |

**References**

* Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.1810.04805>
* Mistral AI. (2023). *Mistral-7B-Instruct-v0.2 model card*. <https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2>
* Meta AI. (2024). *LLaMA 3: Open and efficient foundation language models*. https://ai.meta.com/llama
* Zhang, Y., Jin, R., & Zhou, Z. (2010). Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, *1*(1–4), 43–52. <https://doi.org/10.1007/s13042-010-0001-0>
* Eyben, F., Wöllmer, M., & Schuller, B. (2010). Opensmile. *Proceedings of the 30th ACM International Conference on Multimedia*. <https://doi.org/10.1145/1873951.1874246>
* Pham, G. V., Shancer, M., & Nelson, M. R. (2018). Only other people post food photos on Facebook: Third-person perception of social media behavior and effects. *Computers in Human Behavior*, *93*, 129–140. <https://doi.org/10.1016/j.chb.2018.11.026>
* Wu, C., Li, W., Wang, L., & Zomaya, A. Y. (2020). An evolutionary fuzzy scheduler for multi-objective resource allocation in fog computing. *Future Generation Computer Systems*, *117*, 498–509. https://doi.org/10.1016/j.future.2020.12.019
* Ani Aghababyan, Nicholas Lewkow, and Ryan Baker. 2017. Exploring the Asymmetry of Metacognition. In Proceedings of the Seventh International Learning Analytics & Knowledge Conference (Vancouver, British Columbia, Canada) (LAK '17). Association for Computing Machinery, New York, NY, USA, 115--119. <https://doi.org/10.1145/3027385.3027388>
* Vivian Hsueh Hua Chen and Jeremy Ong. 2016. The rationalization process of online game cheating behaviors. Information, Communication & Society, Vol. 21, 2 (Dec. 2016), 273--287. <https://doi.org/10.1080/1369118x.2016.1271898>
* Maryam Mustafa, Kimia Tuz Zaman, Tallal Ahmad, Amna Batool, Masitah Ghazali, and Nova Ahmed. 2021. Religion and Women's Intimate Health: Towards an Inclusive Approach to Healthcare. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (Yokohama, Japan) (CHI '21). Association for Computing Machinery, New York, NY, USA, Article 232, 13 pages. <https://doi.org/10.1145/3411764.3445605>