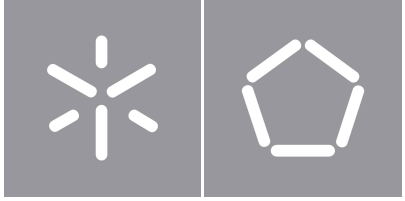


University of Minho
School of Engineering

Rita Mendes Gomes

Artificial Intelligence in Recruitment Management: The Case of AssessBot

september 2024



University of Minho
School of Engineering

Rita Mendes Gomes

Artificial Intelligence in Recruitment Management: The Case of AssessBot

Masters Dissertation
Master's in Informatics Engineering

Dissertation supervised by

Professor Rui Castro Mendes

september 2024

Copyright and Terms of Use for Third Party Work

This dissertation reports on academic work that can be used by third parties as long as the internationally accepted standards and good practices are respected concerning copyright and related rights.

This work can thereafter be used under the terms established in the license below.

Readers needing authorization conditions not provided for in the indicated licensing should contact the author through the RepositóriUM of the University of Minho.

License granted to users of this work:



CC BY

https://creativecommons.org/licenses/by/4.0/

Acknowledgements

Quero agradecer em especial ao João Nuno, pela presença constante e por me dizer sempre todas as verdades difíceis de ouvir. Quero agradecer ao meu leal companheiro que me acompanha há 20 anos, o Berna. Vocês são, sem dúvida, os melhores amigos que eu podia ter. Agradeço-vos pelo apoio, pelo amor e pela amizade que compartilhamos.

Agradeço ao Zé Tolo por todas as aventuras vividas nestes 6 anos de universidade e por me ensinar a não levar a vida tão a sério e da importância de celebrá-la na inconsequência de sermos só tolos.

À Sofia, Paulinha, Bruna, Monteiro, Anabela, Gonçalo, Paulo e Fábio por me porem os pés assentes na realidade.

Quero agradecer aos meus pais, por me darem todos os recursos para concretizar isto. Agradeço também ao meu irmão Diogo, cujas dúvidas discretas sobre o meu sucesso académico me deram o impulso extra para concluir o curso.

Quero também expressar a minha gratidão ao meu orientador, Rui Castro Mendes, pelo apoio e orientação ao longo desta dissertação.

Para além disso, quero agradecer a todas as pessoas que conheci e que se cruzaram comigo ao longo desta jornada, todos, de uma forma ou de outra, ajudaram-me a crescer, a ver várias formas de viver a vida e a construir memórias juntos.

E por último e mais importante, agradeço a mim mesma, por nunca ter desistido, por superar as adversidades, por ter evoluído, por ter usufruído do processo até chegar aqui, sem nunca perder o fio à meada. Esta jornada foi repleta de aventuras, lágrimas, gargalhadas, noites na biblioteca e noites a dançar até o corpo não aguentar. Estou muito orgulhosa daquilo que consegui alcançar e da pessoa que me tornei. A felicidade só é sentida quando compartilhada. Por isso compartilho aqui o quão feliz estou por encerrar este capítulo da minha vida.

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, Braga, september 2024

Rita Mendes Gomes

Abstract

The field of recruitment is traditionally a challenging and complex area, heavily reliant on human judgment. However, the emergence of Artificial Intelligence (AI) is transforming the ways we approach this field. This dissertation presents an exploration of the use of AI in recruitment, with a particular focus on the development of AssessBot, a platform designed to automate and refine the technical evaluation of candidates. Developed in collaboration with Accenture, this platform embodies the convergence of technological advancement and practical Human Resources needs.

AssessBot utilizes generative AI to create personalized technical tests, aiming to improve the performance and complexity of candidate evaluations. This innovative tool aims to significantly expedite the recruitment process, reduce associated costs, and, crucially, mitigate the inherent biases prevalent in traditional hiring practices. This research explores the complete development journey of the platform, with a particular focus on practicality, ethics, and user experience.

Through a critical analysis of various recruitment platforms using AI, the dissertation highlights the contributions and limitations of existing technologies. In doing so, it projects the future advancements of AssessBot, outlining the planned improvements and optimization strategies aimed at enhancing both the platform's functionalities and the user experience.

In summary, this research highlights the potential of AI to revolutionize the field of recruitment, offering perspectives on how these technologies can transform efficiency, equity, and the future of work.

Keywords Artificial Intelligence, Recruitment, Assessment, Human Resources, Generative AI

Resumo

A área do recrutamento é tradicionalmente um campo desafiador e complexo, dependendo significativamente do julgamento humano. Contudo, o surgimento da Inteligência Artificial (IA) está a transformar a forma como abordamos esta área. Esta dissertação apresenta uma exploração do uso da IA em recrutamento, com foco particular no desenvolvimento do AssessBot, uma plataforma projetada para automatizar e refinar a avaliação técnica dos candidatos. Desenvolvida em colaboração com a Accenture, esta plataforma incorpora a convergência do avanço tecnológico e das necessidades práticas de Recursos Humanos.

O *AssessBot* utiliza IA generativa para criar testes técnicos personalizados, visando melhorar o desempenho e a complexidade das avaliações dos candidatos. Esta ferramenta promete acelerar significativamente o processo de recrutamento, diminuir os custos associados e, importante, diminuir as tendências inerentes frequentemente observados nas práticas de recrutamento tradicionais. Esta pesquisa traça todo o caminho até à implementação da plataforma, com ênfase na praticidade, ética e na experiência do utilizador.

Através de uma análise crítica de várias plataformas de recrutamento com recurso a IA, a dissertação destaca as contribuições e as limitações das tecnologias existentes. Com isso, projeta os futuros avanços do *AssessBot*, apresentando as melhorias planeadas e estratégias de otimização destinadas a aperfeiçoar tanto as funcionalidades da plataforma como a experiência do utilizador.

Em suma, esta pesquisa destaca o potencial da IA para revolucionar o campo do recrutamento, oferecendo perspetivas sobre como essas tecnologias podem transformar a eficiência, a equidade e o futuro do trabalho.

Palavras-chave Inteligência Artificial, Recrutamento, Avaliação Técnica, Recursos Humanos, IA Generativa

Contents

1 Introduction **1**

1.1 Motivation and Problem Statement . . . 2
1.2 Objectives . . . 3
1.3 Methodology . . . 3
1.4 Manuscript Outline . . . 6

2 Fundamental Concepts **7**

2.1 Traditional Recruiting . . . 7
2.2 Artificial Intelligence . . . 9
2.3 Machine Learning . . . 10
2.4 Neural networks . . . 11
2.5 Deep Learning . . . 12
2.6 Generative AI . . . 13
2.7 Large Language Models . . . 14
2.8 Vertex AI . . . 16
2.8.1 PaLM2 . . . 16
2.8.2 Gemini . . . 17
2.9 ChatGPT . . . 18
2.10 Comparison between the different generativeAI models used . . . 19
2.10.1 Model Architectures . . . 19
2.10.2 Training Techniques . . . 22
2.10.3 Compliance and Ethical Considerations . . . 25
2.11 Prompt Engineering . . . 26
2.11.1 Benefits of prompt engineering . . . 27
2.11.2 Best practices for prompt engineering . . . 27

2.12 Firebase Realtime Database . . . 28
2.12.0.1 Firebase Authentication . . . 29

3 Review of Artificial Intelligence Methods for Recruitment Processes 31
3.1 Hirevue . . . 31
3.2 Pymetrics . . . 33
3.3 Eightfold . . . 34
3.4 HireEZ . . . 35
3.5 Textio . . . 36
3.6 Paradox . . . 37
3.7 Findem . . . 38
3.8 Humanly.io . . . 39
3.9 HiredScore . . . 40
3.10 Comparing all platforms . . . 41

4 Implementation 43
4.1 Familiarization with VertexAI . . . 43
4.2 Model Training . . . 45
4.2.1 PaLM2 - Text-Bison-32k (latest) . . . 45
4.3 PaLM2 vs GPT 3.5 . . . 49
4.4 Feasibility Analysis . . . 50
4.5 Architecture Solution . . . 57
4.6 Domain Model . . . 59
4.7 Workflow . . . 61
4.8 Database Structure . . . 63
4.8.1 Examples . . . 63
4.8.2 Demands . . . 64
4.8.3 Quizzes . . . 65
4.8.4 TempCandidates . . . 66
4.8.5 TempRecuiters . . . 67
4.8.6 Users . . . 68
4.9 Platform Design . . . 69
4.9.1 Recuiters . . . 69

4.9.2 Candidates . . . 77
4.9.3 Administrator . . . 80
4.10 Script for validation of generated questionnaires . . . 81
4.11 Price estimation and Deployment . . . 82
4.11.1 Google Cloud Run . . . 82
4.11.1.1 Challenges Faced with Google Cloud Run Deployment . . . 85
4.11.2 Google App Engine . . . 86
4.11.3 Google Kubernetes Engine . . . 86

5 Challenges during development **89**
5.1 Differentiating Question Seniority Levels . . . 89
5.2 Authentication Decisions . . . 89
5.3 Code Snippets Generation . . . 90
5.4 Application Bottleneck . . . 92
5.5 Database Security . . . 92
5.6 Database Choice . . . 93
5.7 Deployment Challenges . . . 94

6 Conclusions and future work **95**
6.1 Conclusions . . . 95
6.2 Prospect for future work . . . 96

List of Figures

1 Agile diagram 5

2 Traditional recruitment process. 8

3 Artificial Neural Network vs Deep Neural Network (Mostafa et al., 2020). 13

4 Relation between the concepts. 16

5 Firebase Authentication dashboard. 30

6 VertexAI console. 46

7 Examples fields before refinement. 48

8 Examples fields after refinement. 49

9 System Architecture. 58

10 Domain Model. 60

11 Workflow. 62

12 Database Structure. 63

13 Login page. 69

14 Sign up page. 69

15 Demands list page. 70

16 Create Demand page. 70

17 Demand page. 71

18 List of candidates associated with the demand. 72

19 Quiz parameters. 73

20 Carlos's profile. 74

21 Statistics Dashboard. 75

22 Dashboard for one specif demand. 76

23 My demands page for candidates. 77

24 Quiz instructions. 78

25 Quiz. 79
26 My Demands page for candidates with a closed demand. 79
27 Demands management - Admin page. 80
28 Google Cloud pricing calculator. 82
29 Estimated cost breakdown for Google Cloud Run. 85
30 Estimated cost breakdown for Google Kubernetes using 2 containers. 88

List of Tables

1 Comparison between different models (Team et al., 2024). 25

2 Feasibility Analysis. 54

3 Distribution of Questions Across Technology Areas. 56

4 Detailed analysis of job demands showing input and output character counts across different runs and seniority levels. 83

5 Cost estimation for Google Vertex AI. 83

6 Google Cloud Run pricing details. 84

Chapter 1

Introduction

This manuscript presents the work proposal developed during the final year of the Master's degree in Informatics Engineering at the University of Minho, throughout the 2023-2024 academic year. This period was dedicated to exploring Artificial Intelligence (AI) tools within the context of automated recruitment through the AssessBot project, in collaboration with Accenture. Accenture is a global professional services company specializing in Information Technology (IT) services and consulting, with a strong focus on digital transformation and operational efficiency. Headquartered in Dublin, Ireland, Accenture offers a wide range of services including strategy, consulting, digital, technology, and operations across more than 40 industries. The company is known for its deep expertise in high-performance computing and large-scale systems integration. By leveraging innovative technologies such as artificial intelligence and cloud computing, Accenture assists clients in improving their performance and creating lasting value. Operating in over 120 countries, Accenture is recognized for its extensive global reach and ability to deliver measurable outcomes for its stakeholders. While working on the project, I was based at Accenture's offices in Braga. This partnership aimed to integrate real-world professional service practices with academic research in to enhance recruitment processes.

During this period, we developed and validated AssessBot, an innovative platform designed to automate the technical assessment of candidates in the recruitment process. The goal was to improve the current system for the technical evaluation of candidates, in terms of performance and complexity, through the accurate detection of technical skills required for job positions. AssessBot aimed to use Generative Artificial Intelligence (GenAI) technologies to create and evaluate personalized technical tests, thus facilitating the selection and recruitment process.

The success of AssessBot could mean a considerable advancement in the way IT recruiters evaluate and select candidates, reducing the time needed to find the ideal profile and reducing the costs associated with the recruitment process. All the proposed objectives and tasks were explored in detail in the subsequent chapters of this document.

1.1 Motivation and Problem Statement

In the realm of Human Resources (HR), the department plays a pivotal role in shaping an organization's employee landscape, overseeing recruitment procedures, ensuring the adaptability of chosen applicants to the company, monitoring employee satisfaction, and ultimately bearing responsibility for personnel performance and productivity. In this context, emerges as an indispensable tool to optimize processes and boost productivity in this dynamic sector (Sipahi and Artantaş, 2022).

The current competitive landscape has heightened the quest for talent, making it crucial for companies to efficiently handle recruitment demands while simultaneously reducing associated costs. Particularly during hiring seasons, the recruitment team faces significant pressure to identify and attract the right professionals to the organization (Sen et al., 2023).

Conventional recruitment methods often exhibit bias towards factors such as color, race, and education, raising concerns about the fairness of the process. The integration of and Machine Learning (ML) in recruitment is seen as a transformative force in talent acquisition, driven by the need to establish unbiased hiring practices (Sipahi and Artantaş, 2022).

Automation and efficiency are crucial objectives, given the volume of available data and the potential for mass reach through job descriptions, fueling challenges such as data cleaning and effective management (Sen et al., 2023).

The motivation to explore in recruitment extends beyond operational efficiency to the promotion of fairness, accuracy, and comprehensive candidate evaluation, seeking to build a diverse and robust workforce (Sipahi and Artantaş, 2022).

Considering the limitations of conventional methods, such as bias and inconsistencies in candidate ranking, the automation of information verification becomes an urgent need for HR departments. The anticipated advantages of an AI-based model include speeding up the hiring process, enhancing workflow, standardizing job placement, and reducing costs (Meshram, 2023).

In light of these considerations, this project aims not only to address existing gaps in conventional recruitment but also to propose an AI-driven recruitment model that not only simplifies the hiring process but also promotes principles of equity, efficiency, and comprehensive candidate assessment. This initiative aligns with the goals of operational efficiency and the cultivation of a diverse and strong workforce, essential for ongoing success in a competitive market (Sipahi and Artantaş, 2022).

1.2 Objectives

This dissertation aims to explore and enhance the application of techniques in the hiring and recruitment processes to improve efficiency, fairness, and the overall quality of candidate selection. To achieve this primary goal, the following specific objectives are established:

- **Objective 1**: Evaluate current systems in recruitment.
- **Objective 2**: Define specifications for an AI-enhanced recruitment system.
- **Objective 3**: Integrate into a user-friendly recruitment platform.
- **Objective 4**: Validate the newly developed recruitment system using real-world recruitment scenarios and datasets.

The pursuit of these objectives will enable the answering of pertinent Research Questions (RQs):

- **RQ 1:** How can ML algorithms be tailored to improve the objectivity and efficiency of the recruitment process while minimizing inherent biases?
- **RQ 2:** What are the key design considerations for AI-based recruitment systems to ensure they meet the practical needs of HR professionals and adhere to ethical standards?

1.3 Methodology

During this study, I had the valuable opportunity to undertake a nine-month internship at Accenture, which served as the practical foundation for this thesis. The chosen methodological approach for this study is based on Agile principles, with a specific focus on Scrum, one of the most widely adopted agile methods. This project management framework emphasizes flexibility in adapting to changing requirements, ensuring efficiency in managing costs, scope, and software quality (Hayat et al., 2019).

The Scrum lifecycle involves sprints, which are defined time blocks, typically lasting 2 to 3 weeks, in which teams complete tasks (Chaudhari and Joshi, 2021). The project utilized Azure DevOps, Microsoft's platform for software development management, particularly Azure Boards, and Azure Repos, as essential tools for collaboration and project tracking.

Azure Boards played a pivotal role in planning and tracking project activities. It facilitated the creation and management of work items such as user stories, bugs, and tasks, ensuring they were prioritized,

assigned to team members, and monitored throughout the project lifecycle. Its robust features supported sprint planning, progress tracking, and comprehensive reporting, thereby enhancing project management efficiency.

Azure Repos, on the other hand, served as a Git-based source control service, providing unlimited private repositories for managing the project's source code. It enabled seamless collaboration among team members, allowing concurrent work on new features or bug fixes through branching and facilitating peer code reviews via pull requests. Its integration with Azure Pipelines and Azure Boards ensured a cohesive approach to managing the software development lifecycle.

In Scrum, the process is structured into several essential stages, each with distinct roles and responsibilities. It begins with the creation and prioritization of the Product Backlog, which lists all desired features for the product. This backlog is managed and ranked within Azure DevOps, offering a transparent view of upcoming development tasks for each sprint. The Product Owner, advocating for stakeholder interests, holds the responsibility of maintaining and prioritizing this backlog.

Following the backlog is Sprint Planning, a session where the development team, alongside the Product Owner and Scrum Master, determines which items from the Product Backlog will be addressed in the upcoming sprint. The objective is to select tasks feasible for completion during the Sprint and outline a strategy to achieve the defined goals.

During the sprint, the team holds daily meetings called Daily Scrums. These fast-paced, focused meetings allow the team to discuss progress, identify obstacles, and plan work for the next 24 hours.

At the conclusion of each Sprint, a Sprint Review occurs. During this session, the team presents the completed features to the Product Owner and other stakeholders, receiving feedback and making necessary adjustments to the Product Backlog.

After the Sprint Review, the team conducts a Sprint Retrospective, a meeting to reflect on the sprint's progress, identify what worked well, and what can be improved. The goal is to promote continuous improvement of the team's work and the development process.

Throughout the Scrum process, the Scrum Master plays a pivotal role in ensuring adherence to Scrum principles. They facilitate meetings, remove obstacles, and promote a collaborative and productive work environment.

Lastly, the Product Owner remains actively engaged with the team, representing stakeholder interests, making decisions on Sprint inclusions, and ensuring the product aligns with customer needs.

These defined stages and roles within Scrum provide a structured framework for software development, enabling rapid and frequent delivery of value to customers while encouraging collaboration and adaptability

to evolving business conditions (Hayat et al., 2019).

To facilitate the integration of Azure Boards with the project, we utilized the git commits feature, which allows for quick and direct association between commits and Azure Boards tickets. When making a commit, including the corresponding ticket ID automatically associates the commit with the backlog item. This provided a clear view in Azure Boards, with direct access to the code related to each task, allowing the team to easily track progress and review changes made.

By leveraging Azure DevOps within the Scrum framework, the project benefited from structured project management, streamlined collaboration, and continuous delivery of value to stakeholders. The integration of Azure Boards with git commits further enhanced transparency and traceability, allowing the team to link code changes directly to project tasks and monitor progress effectively.

This approach not only supported efficient software development but also aligned with agile principles of adaptability, collaboration, and customer satisfaction, demonstrating the effectiveness of Agile methodologies in practical project environments.

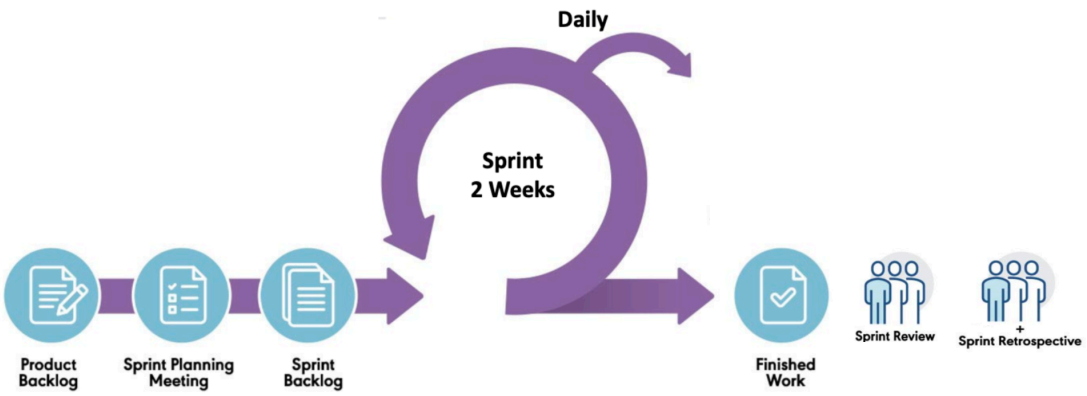


Figure 1: Agile diagram

1.4 Manuscript Outline

This document comprises five chapters, each providing unique insights into how artificial intelligence has reshaped the talent recruitment process. In Chapter 1, the "Introduction", we explored the motivations behind this research, outlined the objectives, and detailed the methodology employed. This chapter set the stage by addressing the challenges faced by hiring practices and advocating for smarter, more efficient solutions.

Chapter 2, "Fundamental Concepts", delves into essential topics such as traditional recruitment methods, AI, ML, neural networks, deep learning (DL), generative AI, Large Language Models (LLMs), VertexAI, etc. This foundational chapter provides the necessary background to understand subsequent discussions on in recruitment.

Chapter 3, "Review of Artificial Intelligence Methods for Recruitment Processes", presents a comprehensive analysis of existing technologies used in recruitment. This review not only highlights technological advancements but also critically examines gaps and opportunities for improvement, with a specific focus on their application in recruitment.

Chapter 4, "Implementation", encompasses critical elements such as model training, system architecture, database structure, platform design and deployment strategies. It provides a detailed walkthrough from the initial stages of model development to the final deployment phase.

Chapter 5, "Challenges During Development", addresses the significant obstacles encountered throughout the development of AI solutions for recruitment. This chapter discusses technical, operational, and strategic challenges, providing insights into overcoming these hurdles effectively

Finally, Chapter 6, "Conclusions and Future Work", concludes the document by summarizing the key findings of the study. It discusses the comparative analysis of various explored technologies and explores the innovative potential of the AssessBot platform in recruitment. Additionally, this chapter outlines future research directions and anticipated developments in the field.

Chapter 2

Fundamental Concepts

In this chapter, I embark on an extensive exploration of foundational concepts in technology, providing crucial insights to contextualize the ensuing case study. I begin by examining traditional recruitment methodologies and then transition to the cutting-edge applications of AI, encompassing ML, neural networks, and deep learning. My goal is to offer a comprehensive understanding of these pivotal components that shape both today's and tomorrow's technological landscapes.

Furthermore, I delve into recent advancements in generativeAI and the specific implementation challenges posed by LLMs. Specifically, I discuss Vertex AI, which includes innovative models like PaLM2 and Gemini, designed to streamline ML workflows and enhance operational efficiency in AI-driven applications.

I also explore ChatGPT, a powerful natural language processing model, and provide a comparative analysis with other prominent models. This comparison covers aspects such as model architecture, training techniques, and ethical considerations, highlighting the nuanced differences that influence their performance and applicability in various contexts.

Lastly, I examine the Firebase Realtime Database, elucidating its functionalities and addressing the complexities involved in managing real-time databases.

2.1 Traditional Recruiting

Recruitment is a strategic procedure employed by organizations to acquire talents that meet the specific needs of their operations. It encompasses a series of activities aimed at identifying qualified candidates and engaging them with the available positions, ultimately selecting the most suitable ones to fill the roles.These positions can range from entry-level roles to leadership functions, depending on the demands and goals of the organization (Parida and Sahu, 2023).

To effectively outline the stages of the traditional recruitment process, it was crucial to first understand the company's journey from identifying the need to hire someone to selecting candidates to fulfill that

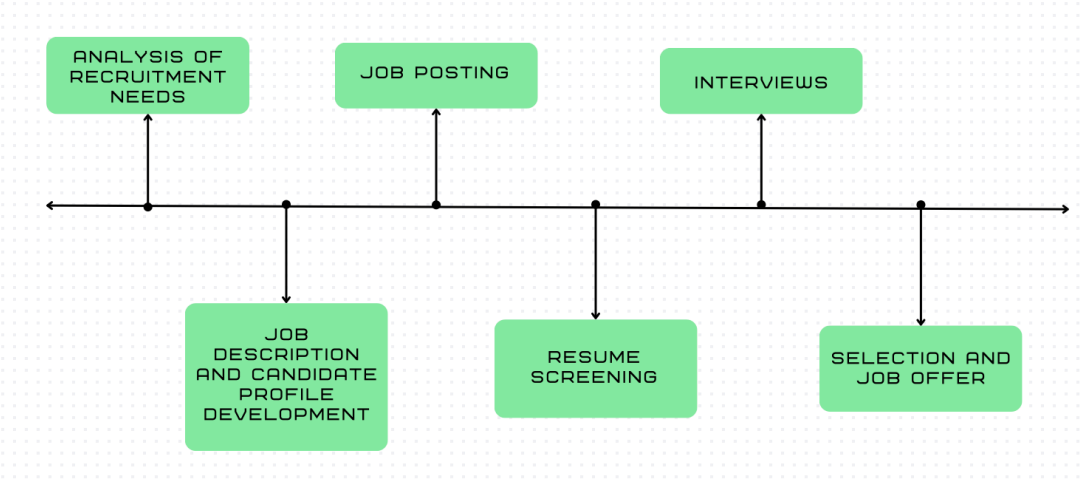


Figure 2: Traditional recruitment process.

requirement. This understanding was achieved through detailed discussions with Accenture regarding the various phases involved in recruitment.

- **Analysis of Recruitment Needs:** This phase involves identifying the staffing needs within the organization. Managers determine which positions need to be filled, as well as the specific skill requirements and qualifications necessary for each role.
- **Job Description and Candidate Profile Development:** Once recruitment needs are identified, managers work on crafting detailed job descriptions and profiles of ideal candidates. This includes specifying responsibilities, required skills, desired experience, and other important qualities for the role.
- **Job Posting:** At this stage, open positions are advertised to the target audience through various channels such as job websites, social media, recruitment agencies, and newspaper ads. The goal is to reach as many qualified candidates as possible.
- **Resume Screening:** Upon receiving applications, recruiters screen resumes to identify candidates who best meet the criteria outlined in the job description. Candidates selected at this stage are invited to proceed to the next phase of the process.
- **Interviews:** Pre-selected candidates are interviewed to assess their skills, experience, cultural fit, and compatibility with the organization. Interviews may vary in format, including individual interviews, panel interviews, or behavioral interviews. Traditional recruitment includes a technical

assessment through specific questions during the interview process. At this stage, AssessBot can be integrated to conduct more detailed technical evaluations.

- **Selection and Job Offer:** Following the interviews, recruiters select the most suitable candidates for the position and extend a formal job offer. This offer includes details regarding salary, benefits, and other employment conditions.

Recent studies have shed light on the nuances and challenges inherent in conventional recruitment practices. According to Khan, Hussain, and Ahmad in their article "Revolutionizing Human Resource Management: The Transformative Impact of Artificial Intelligence (AI) Applications," the pivotal role of human interaction in traditional recruitment is emphasized, highlighting the formation of unique bonds between recruiters and candidates through interpersonal touch. They acknowledged the effectiveness of conventional methods, such as formal interviews, in facilitating communication and minimizing misinterpretations, thereby fostering a conducive environment for candidates. However, concerns were raised regarding the time-intensive nature of these approaches, leading to prolonged waiting periods due to intricate recruitment procedures. Moreover, experts voiced apprehension regarding the prevalence of biases and discrimination within traditional recruitment practices, citing age, gender, heritage, and organizational biases as significant impediments to fostering diversity and enhancing overall team and company performance (Khan et al., 2023).

Furthermore, the process of hiring a person typically takes around 1 to 3 months, or even longer in some cases, as observed by Accenture. These findings underscore the limitations and challenges associated with traditional recruiting methods, prompting a critical reassessment of organizational recruitment approaches. Against this backdrop, the emergence of in recruiting assumes heightened significance as a potential solution to address these inherent shortcomings.

2.2 Artificial Intelligence

Artificial Intelligence refers to the field of study focused on creating intelligent agents capable of emulating human actions and problem-solving without explicit programming, combining human intelligence with machine intelligence (Kumar et al., 2021).

At its essence, encompasses a spectrum of subfields, such as ML, neural networks, and natural language processing, each contributing to its multifaceted capabilities, as we'll explore in subsequent sections.

In today's fast-paced business environment, deployment has surged, bringing about transformative

changes across diverse sectors, including talent acquisition. However, amidst its promises, AI-driven recruitment encounters its set of challenges, which necessitate a thorough examination.

One major concern is the risk of algorithmic bias, wherein systems may inadvertently perpetuate discriminatory practices present in historical data. Additionally, privacy concerns arise regarding the collection and utilization of sensitive candidate information. Moreover, ensuring the transparency and interpretability of models poses a significant challenge, particularly in highly regulated industries. Addressing the ethical and legal implications associated with the use of in recruitment is imperative. Organizations must prioritize fairness, equity, and inclusivity throughout the recruitment lifecycle, guarding against algorithmic bias and discrimination. Compliance with data protection regulations is essential to safeguard candidate privacy and confidentiality (Mhatre, 2023).

Despite these challenges, the integration of in recruitment processes has emerged as a transformative force, offering numerous benefits to organizations. AI-powered solutions can automate repetitive tasks such as resume screening, candidate sourcing, and interview scheduling, thereby saving time and resources for recruiters. Furthermore, algorithms can analyze large datasets to identify top talent, improve candidate matching, and predict future job performance with greater accuracy (Mitsunaga, 2023).

2.3 Machine Learning

ML is a subset of artificial intelligence dedicated to crafting algorithms and models capable of learning and making decisions based on data. Unlike traditional programming, where instructions are explicitly written for a program to execute a particular task, ML models derive insights from examples and available data. The input, typically in the form of a dataset with a variety of data, serves as the foundation for these algorithms. This approach grants models with greater flexibility, enabling them to address complex tasks that may be challenging to articulate through predefined rules (Kumar et al., 2021).

There are different types of learning in ML, each with its own characteristics and applications:

- **Supervised Learning** relies on labeled data for prediction, where algorithms are trained to map input data to output labels. This method encompasses classification, which assigns data to predefined categories, and regression, which examines relationships between variables. For example, a supervised learning model can be trained to recognize patterns in images and classify the images into different categories. Examples of supervised learning algorithms include: Linear Regression, Logistic Regression, Support Vector Machines (SVMs), Decision Trees and Random Forests (Jha et al., 2021).

- **Unsupervised Learning**, on the other hand, analyzes unlabeled data to identify patterns or structures without predefined categories. This type of learning is often used in clustering tasks, where the goal is to group similar data together. Examples of unsupervised learning algorithms include: K-means Clustering and Hierarchical Clustering (Jha et al., 2021).

- **Reinforcement Learning** optimizes behavior through trial and error, interacting with an environment to maximize cumulative rewards. It involves positive reinforcement to encourage desired behavior and negative reinforcement to avoid unfavorable conditions. This type of learning is commonly used in applications such as gaming and robotics, where the model can learn to perform complex tasks through trial and error. Examples of reinforcement learning algorithms include Q-Learning (Jha et al., 2021).

Training ML models involves several steps, starting with data preparation. This includes cleaning, transforming, and normalizing the data to ensure it is suitable for model training. Once the data is ready, the next step is to select the appropriate ML algorithm for the task at hand. After the algorithm is selected, the next step is to train the model using the available data. During training, the model adjusts its parameters to minimize the difference between the model's predictions and the actual labels in the training data. Once the model has been trained, it's important to evaluate its performance using methods such as cross-validation and evaluation metrics to ensure it's making accurate and useful predictions (Nithya and others., 2023).

In the 2.10.2 section, we will further explore these types of learning.

2.4 Neural networks

Neural networks, or Artificial Neural Networks (ANNs), emulate the structure and function of the human brain and are a fundamental class of ML algorithms. They comprise interconnected nodes, known as neurons, organized in layers. Each neuron processes input signals via an activation function, generating an output signal. The connections between neurons are weighted, enabling the network to learn from data by adjusting these weights during training.

Typically, neural networks consist of an input layer, which receives input data, one or more hidden layers where data processing occurs, and an output layer, which produces predictions. The network's architecture can vary in complexity based on the task at hand, with the number of neurons and layers adjusted accordingly (Abiodun et al., 2018).

There are different types of Neural Networks:

- **Feedforward Neural Networks (FNNs):** Information travels in one direction from input to output (Abiodun et al., 2018).
- **Recurrent Neural Networks (RNNs):** They have connections that form cycles, allowing them to exhibit temporal dynamic behavior, making them suitable for sequence data (Teng et al., 2023).
- **Convolutional Neural Networks (CNNs):** Designed to process grid-like data, such as images, using shared weights and pooling layers to extract features (Teng et al., 2023).
- **Generative Adversarial Networks (GANs):** Comprising two neural networks (generator and discriminator) competing against each other to generate realistic data. In GANs, the generator creates samples and the discriminator tries to distinguish between real and generated samples (Teng et al., 2023). It is worth noting that the focus of this work will be on GANs, as we will delve into later sections.

Neural networks are trained using optimization algorithms such as gradient descent and backpropagation. During training, the network adjusts the weights of connections between neurons to minimize a loss function. Performance evaluation metrics include accuracy, precision, and recall (Abiodun et al., 2018).

Despite their effectiveness, neural networks face significant challenges and limitations. Obtaining large amounts of labeled training data can be costly and challenging, particularly for tasks with limited data availability. Additionally, neural networks can exhibit computational complexity and long training times, especially with large-scale datasets or complex architectures. Overfitting is another common issue, where the model memorizes training data instead of generalizing to new data, leading to poor performance on unseen data. Gradient vanishing or exploding during training can also hinder learning and convergence. Furthermore, the interpretability and explainability of neural networks are challenging due to their complex inner workings, making it difficult to trust their predictions in critical applications (Abiodun et al., 2019).

2.5 Deep Learning

Deep Learning is a subfield of ML that focuses on training deep neural networks, which are characterized by having multiple hidden layers between the input and output layers. The primary distinction between traditional neural networks and deep learning lies in the depth of the network architecture (Kumar et al., 2021).

This deep architecture allows deep neural networks to create intricate and hierarchical representations of data. The presence of multiple hidden layers facilitates the learning of increasingly abstract and

complex features from input data. This hierarchical feature learning process is particularly advantageous for handling unstructured data types such as images, audio signals, and text, where automated feature extraction is crucial for precise analysis and processing (Mostafa et al., 2020).

While traditional neural networks can also have multiple hidden layers, deep learning extends this capability with significantly more layers, as illustrated in the figure below:

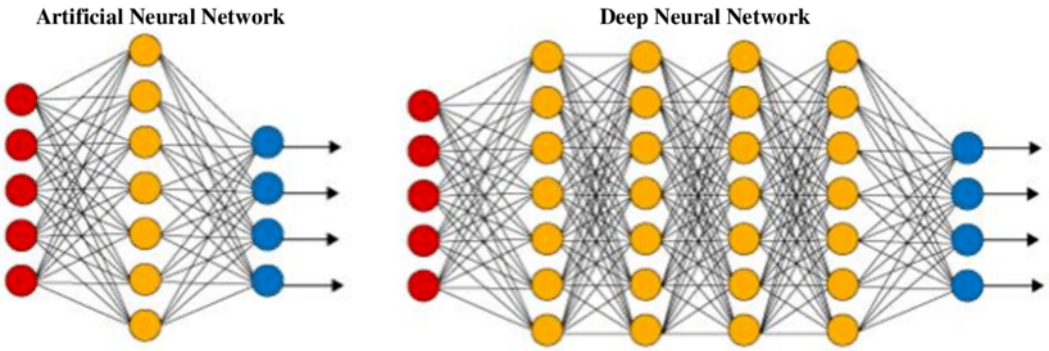


Figure 3: Artificial Neural Network vs Deep Neural Network (Mostafa et al., 2020).

Despite its capabilities, deep learning faces challenges such as the need for large amounts of labeled data, computational complexity, and model interpretability. Some experts caution that these challenges may not be solely addressed by increasing the number of neurons (Zhu et al., 2017).

2.6 Generative AI

Generative AI, or Generative Artificial Intelligence, is a subfield within the broader domain of Deep Learning. It encompasses a class of algorithms and models designed to generate new content that resembles or is derived from existing data patterns. This capability is particularly powerful in fields such as image synthesis, text generation, music composition, and more.

generativeAI relies on various techniques to create new content. Among these, GANs and Variational Autoencoders (VAEs) are prominent:

- **GANs:** Introduced by Goodfellow et al. in their 2014 paper titled "Generative Adversarial Nets", GANs consist of two neural networks, a generator and a discriminator, trained simultaneously in a game-theoretic setting. The generator learns to create realistic samples from noise, while the discriminator learns to distinguish between real and generated samples. The iterative training process, where the generator tries to fool the discriminator and the discriminator aims to catch the generator, results in the generation of high-quality synthetic data (Goodfellow et al., 2014).

- **Variational Autoencoders (VAEs):** VAEs are a type of generative model introduced by Kingma and Welling in their 2013 paper titled "Auto-Encoding Variational Bayes". VAEs are designed to learn a probabilistic latent space from the input data, enabling the generation of new data samples that are similar to the training data. VAEs consist of two main components: an encoder and a decoder. The encoder maps the input data to a latent space, typically assumed to follow a Gaussian distribution. The decoder then reconstructs the input data from the latent space. This approach allows VAEs to generate new samples by sampling from the learned latent distribution (Kingma and Welling, 2013).

Despite its capabilities, generativeAI faces several challenges. Ensuring the quality and diversity of outputs is a significant issue, as it is challenging to generate content that is realistic, diverse, and aligned with user expectations. Ethical considerations also arise, including the amplification of biases and the potential misuse of generated content, which raise important ethical concerns that require careful management (Bender et al., 2021). Additionally, training and inference in generativeAI often demand substantial computational resources, which can limit scalability and accessibility.

2.7 Large Language Models

LLMs represent an advanced class of artificial intelligence systems trained to understand and generate natural language coherently and contextually relevant. These models utilize deep neural networks fueled by vast amounts of textual data to learn intricate linguistic patterns and provide accurate responses across various contexts (Luo et al., 2023).

LLMs undergo a two-phase training process: pretraining on large text corpora to learn general language representations, followed by fine-tuning on specific tasks or domains to enhance performance and adaptability. Two prominent LLMsarchitectures include Generative Pre-trained Transformer (GPT) and BERT (Bidirectional Encoder Representations from Transformers). While both architectures employ transformer-based models, they differ in their training objectives and applications.GPTfocuses on generating coherent and contextually relevant text, making it ideal for tasks such as text generation and completion. In contrast, BERT adopts a bidirectional approach to language modeling, enabling it to understand context and semantics in both directions, which is beneficial for tasks like language understanding and question answering (Brown et al., 2020).

Despite their impressive capabilities, LLMs also face challenges and limitations. Concerns include biases in training data, privacy implications, and the substantial computational resources required for training and deployment (Abdullah et al., 2022). Additionally, LLMs may experience **hallucinations**,

generating outputs that appear plausible but are factually incorrect or irrelevant due to biases in training data or limitations in the model's architecture. These hallucinations arise from several factors, including biases in data lack of real-world context understanding, or inadequacies in the training process (Alkaissi and McFarlane, 2023).

Some common examples of hallucinations include:

- **Incorrect predictions:** An model may predict that an event will occur when it is unlikely to happen. For instance, an model predicting rain tomorrow despite clear weather forecasts (Alkaissi and McFarlane, 2023).
- **False positives:** When working with an model, it may identify something as being a threat when it is not. For example, an model that is used to detect fraud may flag a transaction as fraudulent when it is not (Alkaissi and McFarlane, 2023).
- **False negatives:** An model may fail to identify something as being a threat when it is. For example, an model failing to detect a cancerous tumor in medical diagnostics (Alkaissi and McFarlane, 2023).

Addressing hallucinations in models requires using diverse and representative training data, applying regularization techniques during training to prevent overfitting, and fine-tuning models to specific tasks or domains. Strategies to prevent these hallucinations include restricting prediction ranges, ensuring training with pertinent and specific data (such as medical images for healthcare applications), and incorporating user feedback to align outputs with expectations. It is essential to properly train and test systems with diverse datasets and monitor them using methods like human evaluation or anomaly detection (Alkaissi and McFarlane, 2023).

Effectively managing hallucinations in LLMs is crucial for their ethical and reliable deployment across different domains. By understanding their causes and implementing comprehensive mitigation strategies, developers can improve the accuracy and trustworthiness of LLMs outputs in real-world scenarios.

In this study, we will delve into three prominent LLMs: ChatGPT, Gemini, and PaLM2. In the next section, we will begin by introducing each of these models in detail, followed by a comprehensive comparison focusing on their model architectures, training techniques, and ethical considerations.

Next I present an illustrative image demonstrating the relationship between the concepts discussed in the previous sections.

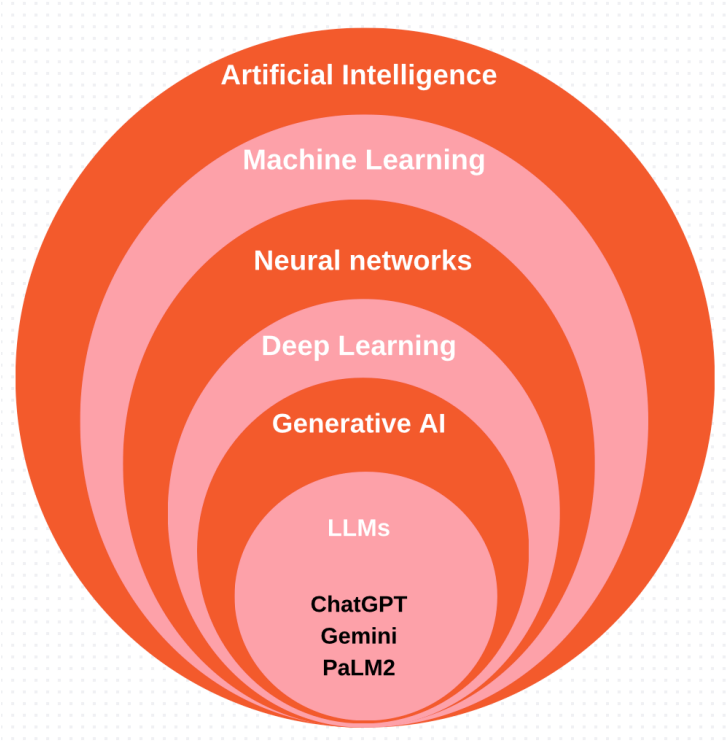


Figure 4: Relation between the concepts.

2.8 Vertex AI

Accenture chose VertexAI as the preferred platform for this project. VertexAI was a unified ML platform provided by Google Cloud that allowed developers and data scientists to build, deploy, and manage ML models at scale. This platform served as the foundation for the development of this project, offering a comprehensive suite of tools and services to streamline the entire ML lifecycle, including data preparation, model training, deployment, and monitoring (Katti et al., 2022). Accenture found VertexAI particularly well-suited for handling LLMs such as PaLM2 and Gemini, providing robust capabilities to leverage these advanced models effectively in real-world applications.

2.8.1 PaLM2

Released by Google on May 23, 2023, PaLM2 represents a significant advancement over its predecessor, PaLM, adopting the Pathways architecture, an extension of the Transformer model. This model was meticulously trained on an extensive and diverse dataset encompassing human and programming languages, mathematical equations, scientific papers, and web content. With a staggering 340 billion parameters trained on 3.6 trillion tokens, PaLM2 marks a milestone in large-scale training, utilizing up to 6144 chips

in the largest TPU-based system configuration employed for training to date (Gong et al., 2023; Chowdhery et al., 2022).

Furthermore, Google has introduced the VertexAI PaLM Application Programming Interface (API) to streamline the integration of PaLM2 models into diverse applications. This API enables users to test, customize, and deploy specific instances of PaLM2 models tailored to their unique requirements, facilitating further fine-tuning to optimize performance (Google Cloud, 2024a).

Throughout the initial three months of this study, PaLM2 served as the primary model under investigation, providing a robust foundation for exploring its capabilities and applications in various contexts, including its utilization and training to assess its viability in generating effective questionnaires for evaluating job candidates during the recruiting process.

2.8.2 Gemini

Developed by Google, Gemini was officially released on December 6, 2023. Unlike other LLMs, Gemini distinguishes itself with its multimodal capabilities, enabling simultaneous processing of diverse data types such as text, images, audio, video, and code (Team et al., 2024).

Gemini comprises three distinct models, Gemini Ultra, Gemini Pro, and Gemini Nano, each tailored for specific tasks. Gemini Ultra, optimized for highly complex tasks, exhibited superior performance compared to GPT-4, while Gemini Pro surpassed GPT-3.5. Notably, Gemini Ultra was the first language model to outperform human experts on the 57-subject Massive Multitask Language Understanding (MMLU) test, achieving a score of 90% (Team et al., 2024). Gemini Pro was made available to Google Cloud customers on VertexAI on December 13, 2024. We use Gemini Pro 1.5 in our study due to its advanced features and robust performance.

Built upon Transformer decoders, Gemini models leverage advancements in architecture and model optimization to facilitate stable training at scale and optimized inference on Tensor Processing Units (TPUs). With a context length support of up to 32k tokens, Gemini models employ efficient attention mechanisms, further enhancing their performance in multimodal tasks and expanding the capabilities of large language models (McIntosh et al., 2023).

Starting in January, we began using Gemini because of its advanced features and promising results. By incorporating Gemini into our study, we aimed to leverage its cutting-edge features and evaluate its performance in comparison to other models like ChatGPT and PaLM2.

Gemini, developed by Google, leverages Google's extensive index, enabling unparalleled real-time analysis of data. Its rapid and precise identification of latent trends, competitor tracking, and comprehension

of evolving dynamics provide researchers with a distinct advantage. Gemini's proficiency in processing extensive datasets enables it to uncover patterns and trends that might elude human analysts, thereby facilitating well-informed decisions and identifying areas for enhancement. Additionally, Gemini's logical orientation makes it suitable for various analytical tasks. Leveraging Google's extensive knowledge repository, Gemini prioritizes factual accuracy and proficient information retrieval, excelling in processing, summarizing, and deriving insights from vast data pools. This attribute positions Gemini as a valuable asset for research endeavors and decision-making processes.

2.9 ChatGPT

Prior to our exploration of Gemini and following the training of the PaLM2 model, we undertook an investigation using the GPT-3.5 model, which was publicly released in November 2022, to compare its performance with PaLM2, particularly in generating more engaging questionnaires. This comparative analysis will be elaborated further in chapter 4. Now, let's delve into ChatGPT in more detail.

ChatGPT, short for Chat Generative Pre-trained Transformer, is an advanced natural language processing (NLP) model developed by OpenAI. Based on the GPT architecture, it represents a fine-tuned iteration optimized for conversational applications. Initially introduced in 2018, GPT has evolved through several versions (Abdullah et al., 2022).

ChatGPT has a substantial knowledge base derived from its training data, but it operates independently and is not connected to the internet for real-time updates beyond its training period, typically up to 2021. User interactions with the system are not confidential and may be reviewed to improve the model's performance. Users are cautioned against sharing confidential or personal information during interactions. The system also notifies users of potential inaccuracies or inappropriate content in ChatGPT responses. Despite these constraints, ChatGPT can retain and build upon previous dialogues within a session, allowing users to continue and revisit ongoing conversations.

The GPT-3.5 model, which includes ChatGPT, boasts 175 billion parameters. In terms of tokens, the free preview of ChatGPT is limited to approximately 3000 words (or 4000 tokens). These parameters and tokens represent the weights, biases, and input/output capabilities learned by the model during training, contributing to its capacity to generate highly accurate and contextually relevant text across a wide range of applications (Gupta et al., 2023).

ChatGPT serves a variety of purposes, including writing code, analyzing data, formulating ideas, editing and producing natural language content, answering questions, and even writing poetry. These capabilities

demonstrate the versatility and utility of ChatGPT across different domains, showcasing its potential to enhance productivity, creativity, and communication in various contexts (Naik et al., 2023).

Despite its impressive capabilities, ChatGPT, like other LLMs, is not without limitations. It faces challenges such as bias in training data, occasional generation of nonsensical output (hallucinations), and susceptibility to interpretational variations in input phrasing. The model tends to overuse vocabulary and repeat words or phrases, an issue originating from biases in the training data and common concerns about over-optimization. Ensuring the model's ethical use and preventing the propagation of inappropriate or harmful content remain critical considerations in its application (Bahrini et al., 2023).

ChatGPT, an innovation from OpenAI, excels in providing nuanced qualitative insights; it adeptly analyzes feedback, discerns sentiment, and grasps subtle shifts in attitudes. ChatGPT's adaptability and conversational skills enhance user interactions and creative content production, making it highly useful for interactive applications. ChatGPT excels in rapid idea generation, making it invaluable for brainstorming sessions and suggesting unconventional approaches. With its adaptive nature and conversational flair, ChatGPT finds suitability in situations where mimicking human interaction holds significance. Its capabilities extend to generating text resembling human speech, crafting dialogues, language translation, and creating diverse forms of creative content.

2.10 Comparison between the different generativeAI models used

Now that we have acquired knowledge and a brief overview of the three models used—PaLM2, Gemini, and ChatGPT—let's delve into detailed discussions on their model architecture, training techniques, and ethical considerations.

2.10.1 Model Architectures

All three models - ChatGPT, PaLM2, and Gemini - are built upon the Transformer architecture, that consists of multiple layers of encoder blocks, each equipped with self-attention mechanisms and feedforward neural networks.

The Transformer model, introduced in the paper "Attention is All You Need" by Vaswani et al. (2023) in 2017, revolutionized neural network architectures by relying solely on self-attention mechanisms. Before the advent of Transformers, traditional models like RNNs or RNNs relied on recurrence or convolutions for sequence transduction tasks. These models typically used an encoder-decoder architecture connected

through an attention mechanism to capture relationships within sequences. Transformers introduced a groundbreaking approach by eliminating recurrent and convolutional layers entirely. Instead, they utilize multi-headed self-attention mechanisms across multiple layers of encoder blocks. This architectural shift allows Transformers to process entire input sequences in parallel, rather than sequentially. This parallel processing capability significantly accelerates training speed and enhances the model's ability to capture long-range dependencies and contextual information efficiently (Vaswani et al., 2023; Evstropov et al., 2023).

As discussed in 2.7, the Transformer architecture serves as the foundation for both GPT and BERT. These models adapt the Transformer's encoder framework with varying approaches to self-attention. GPT focuses on generating coherent and contextually relevant text using a unidirectional self-attention approach. Here, each token in the sequence attends only to preceding tokens. Models like ChatGPT excel in tasks such as text generation, completion, and other forms of language generation. In contrast, BERT utilizes a bidirectional encoder where each token can attend to both preceding and subsequent tokens. This bidirectional self-attention enables BERT models, including variants like PaLM2, to capture context and semantics from both directions. BERT models are well-suited for tasks such as language understanding, text comprehension, machine translation, and question answering.

Let's now delve into the key components of the Transformer architecture:

- **Self-Attention Mechanism:** Self-attention, or intra-attention, is a mechanism that allows each word in the input sequence to evaluate its significance relative to every other word, dynamically adjusting based on contextual relationships. This feature enhances scalability and model capacity while enabling effective capture of comprehensive dependencies across input sequences, thereby improving overall understanding and representation of the input text (Buscemi and Proverbio, 2024; Evstropov et al., 2023).

- **Encoder-Decoder Structure:**

  - **Encoder:** Composed of a stack of six identical layers, each containing a multi-head self-attention mechanism and a fully connected feed-forward network. Residual connections and layer normalization are employed around each sub-layer to facilitate training (Buscemi and Proverbio, 2024).

  - **Decoder:** Similar to the encoder but with an additional multi-head attention sub-layer that performs attention over the encoder's output. This ensures that the decoder can attend to all positions in the input sequence. The decoder also includes masking to prevent it from

attending to future positions, preserving the auto-regressive property of the model (Buscemi and Proverbio, 2024).

- **Multi-Head Attention:** This mechanism is used in three ways:
  - **Encoder-Decoder Attention:** Queries from the decoder attend to the encoder's output (Buscemi and Proverbio, 2024).
  - **Encoder Self-Attention:** Each position in the encoder attends to all positions in the previous layer of the encoder (Buscemi and Proverbio, 2024).
  - **Decoder Self-Attention:** Each position in the decoder attends to all positions up to and including that position in the decoder, ensuring predictions depend only on known outputs (Buscemi and Proverbio, 2024).
- **Feed-Forward Networks:** Each layer in both the encoder and decoder contains a fully connected feed-forward network applied to each position separately. The FNN in each layer applies non-linear transformations to further process the information derived from the self-attention mechanism. This network is responsible for generating the final representations of the input text, which are subsequently harnessed for generating output responses (Buscemi and Proverbio, 2024).

Transformers offer several computational benefits:

- **Parallelization:** Transformers utilize self-attention mechanisms that allow for parallel processing of input sequences. Unlike traditional models such as RNNs or CNNs, which process inputs sequentially or in a limited parallel manner, Transformers can process all tokens in the sequence simultaneously. Each token can attend to all other tokens in the sequence, which is computationally efficient and reduces the time complexity associated with sequential processing (Buscemi and Proverbio, 2024).
- **Path length reduction:** Self-attention mechanisms in Transformers reduce the maximum path length between input and output positions. In traditional models like RNNs, the information flow from one part of the sequence to another can degrade over time due to recurrent connections or convolutions. Transformers mitigate this issue by allowing each token to directly attend to any other token in the sequence, regardless of distance. This capability enhances the model's ability to capture long-range dependencies effectively without the information degradation typical in sequential models (Buscemi and Proverbio, 2024).

- **Computational complexity:** Transformers are capable of efficiently handling large datasets and complex tasks. They achieve this with fewer sequential operations compared to traditional models like RNNs and CNNs, which often require extensive processing steps to propagate information across the network. This makes Transformers less computationally intensive, enabling faster training and more efficient processing of data (Buscemi and Proverbio, 2024).

2.10.2 Training Techniques

The performance of language models is critically influenced by their training methodologies, the scale of their training datasets, and the number of parameters. In this section, we delve into the distinct training methodologies and capabilities of Palm2, Gemini, and ChatGPT, highlighting their approaches to learning language representations and optimizing performance for diverse applications.

Through pre-training, the model learns to predict the next word in a given text sequence based on the context that precedes it. These capacities enable the model to learn the statistical patterns and structural characteristics of human language, modelling extensive dependencies and ideally capturing the contextual nuances present in the text (Buscemi and Proverbio, 2024).

The three models are trained on massive datasets, boasting significant parameters and extensive training data. This large-scale training enables the models to generalize effectively across diverse inputs and contexts, ensuring the generation of high-quality outputs. For instance, ChatGPT 3.5 boasts approximately 10^{11} parameters, indicative of the scale at which these models operate (Buscemi and Proverbio, 2024).

Gemini undergoes training on a significant scale using Google's vast and diverse dataset. It employs a robust pre-training strategy followed by fine-tuning for specific conversational and multimodal tasks. Initial pre-training on Gemini establishes a strong foundation, enabling the model to capture complex contextual nuances and optimize for various specialized tasks. Gemini's fine-tuning strategy incorporates advanced optimization techniques to adapt the model for improved accuracy and efficiency in generating responses across real-world scenarios. The richness and diversity of training data at Google contribute to Gemini's effective generalization and high-quality output generation (Team et al., 2024).

PaLM2, on the other hand, leverages a diverse pre-training corpus that includes a substantial amount of non-English monolingual and parallel data, enhancing its capability for multilingual tasks such as translation and multilingual question answering. This approach allows PaLM2 to capture linguistic nuances across multiple languages and cultures, benefiting from its training on parallel multilingual data. PaLM2's extended context length capability supports tasks that require deep understanding of long passages or dialogue, further enhancing its performance in tasks requiring long-range reasoning and comprehension

(Anil et al., 2023).

While Gemini prioritizes logical structuring for precise data analysis and ChatGPT explores qualitative insights comprehensively, PaLM2 focuses on leveraging its extensive multilingual training corpus for robust multilingual understanding and performance.

Effective optimization techniques are crucial for training sophisticated models.

- **Supervised Learning:** Supervised learning utilizes labeled datasets to guide models towards accurate predictions. Recent advancements have focused on developing sophisticated loss functions and regularization techniques to enhance the performance and generalization capabilities of supervised learning models, ensuring robustness across a wide range of tasks and data types (McIntosh et al., 2023).

- **Unsupervised Learning:** Unsupervised learning plays a crucial role in by uncovering patterns within unlabeled data. Significant advancements have been made with the introduction of autoencoders and GANs, expanding unsupervised learning's applicability and enabling sophisticated data generation and representation learning capabilities. These innovations are vital for understanding and leveraging complex structures inherent in unstructured datasets, highlighting the versatility and depth of unsupervised learning techniques (McIntosh et al., 2023).

- **Transfer Learning:** Transfer learning emphasizes versatility and efficiency in training by allowing models to apply knowledge acquired from one task to different yet related tasks. This reduces the need for large labeled datasets for each new task. By utilizing pre-trained networks, transfer learning streamlines the training process, enabling efficient fine-tuning for specific applications. This approach enhances adaptability and performance across diverse tasks, particularly beneficial in scenarios where acquiring extensive labeled data is impractical or unfeasible (McIntosh et al., 2023). Both Gemini and ChatGPT employ transfer learning, where a pre-trained model is fine-tuned with more specific data or tasks. This approach allows the models to leverage previously acquired knowledge, making the training process more efficient and enhancing the models' performance on specialized tasks.

- **Reinforcement Learning Human Feedback (RLHF):** Both ChatGPT and Gemini are fine-tuned using RLHF, where human feedback is utilized to guide the model toward more desirable and useful outputs. This technique improves the quality of responses and the accuracy of generated content by incorporating human insights into the training process (Naik et al., 2023).

To ensure high-quality responses, Gemini models undergo rigorous training data filtering and enrichment processes, adhering to Google DeepMind's best practices on data enrichment. These practices include fair compensation for data enrichment workers to maintain ethical standards (Team et al., 2024).

Instruction tuning efforts focus on three key behaviors to enhance real-world scenario responses:

- Attribution: Gemini produces responses faithful to given contexts, including summarizing user-provided sources, generating citations, and transforming sources into desired outputs (Team et al., 2024).
- Closed-Book Response Generation: Gemini avoids hallucinating incorrect information when provided with fact-seeking prompts without given sources (Team et al., 2024).
- Hedging: Gemini acknowledges unanswerable prompts without hallucination, encompassing scenarios with false-premise questions or answers not derivable from the given context (Team et al., 2024).

In a comprehensive comparison between Gemini Pro, ChatGPT-3.5, and PaLM2, the results reveal that Gemini Pro generally delivers superior or competitive performance across a range of tasks evaluated. Gemini Pro is particularly robust on academic tasks such as MMLU, where it scores 79.13%, outperforming GPT-3.5 (70%) and slightly ahead of PaLM2 (78.4%). On school math problems (GSM8K), Gemini Pro scores 86.5%, again significantly outperforming GPT-3.5 (57.1%) and also PaLM2 (80.0%). However, on more general math tasks (MATH), Gemini Pro's performance (32.6%) is slightly lower than that of GPT-3.5 (34.1%) and PaLM2 (34.4%). In code evaluation tasks (HumanEval), Gemini Pro leads with 67.7%, compared to GPT-3.5's 48.1%. It excels in code generation (Natural2Code) with 69.6%, while GPT-3.5 scores 62.3%. PaLM2 does not have data available for these specific tasks. In reading comprehension and arithmetic tasks (DROP), PaLM2 leads with 82.0%, followed by Gemini Pro (74.1%) and GPT-3.5 (64.1%). In common sense tasks (HellaSwag), the three models perform similarly, with PaLM2 achieving 86.8%, GPT-3.5 with 85.5%, and Gemini Pro with 84.7%. In machine translation (WMT23), PaLM2 slightly outperforms Gemini Pro, with 72.7% versus 71.7%, and GPT-3.5 has no data available.

In terms of overall performance, Gemini Pro proves to be a robust and consistent model, frequently outperforming GPT-3.5 and effectively competing with PaLM2. In addition, Gemini Ultra stands out by outperforming GPT-4 in several tasks, solidifying the Gemini family's position as the performance leader.

	Gemini Ultra	Gemini Pro	GPT-4	GPT-3.5	PaLM 2-L
MMLU Multiple-choice questions in 57 subjects (professional & academic) (Hendrycks et al., 2021a)	**90.04%** CoT@32*	79.13% CoT@8*	87.29% CoT@32 (via API**)	70% 5-shot	78.4% 5-shot
	83.7% 5-shot	71.8% 5-shot	86.4% 5-shot (reported)		
GSM8K Grade-school math (Cobbe et al., 2021)	**94.4%** Maj1@32	86.5% Maj1@32	92.0% SFT & 5-shot CoT	57.1% 5-shot	80.0% 5-shot
MATH Math problems across 5 difficulty levels & 7 subdisciplines (Hendrycks et al., 2021b)	**53.2%** 4-shot	32.6% 4-shot	52.9% 4-shot (via API**)	34.1% 4-shot (via API**)	34.4% 4-shot
			50.3% (Zheng et al., 2023)		
BIG-Bench-Hard Subset of hard BIG-bench tasks written as CoT problems (Srivastava et al., 2022)	**83.6%** 3-shot	75.0% 3-shot	83.1% 3-shot (via API**)	66.6% 3-shot (via API**)	77.7% 3-shot
HumanEval Python coding tasks (Chen et al., 2021)	**74.4%** 0-shot (IT)	67.7% 0-shot (IT)	67.0% 0-shot (reported)	48.1% 0-shot	—
Natural2Code Python code generation. (New held-out set with no leakage on web)	**74.9%** 0-shot	69.6% 0-shot	73.9% 0-shot (via API**)	62.3% 0-shot (via API**)	—
DROP Reading comprehension & arithmetic. (metric: F1-score) (Dua et al., 2019)	**82.4** Variable shots	74.1 Variable shots	80.9 3-shot (reported)	64.1 3-shot	82.0 Variable shots
HellaSwag (validation set) Common-sense multiple choice questions (Zellers et al., 2019)	87.8% 10-shot	84.7% 10-shot	**95.3%** 10-shot (reported)	85.5% 10-shot	86.8% 10-shot
WMT23 Machine translation (metric: BLEURT) (Tom et al., 2023)	**74.4** 1-shot (IT)	71.7 1-shot	73.8 1-shot (via API**)	—	72.7 1-shot

Table 1: Comparison between different models (Team et al., 2024).

2.10.3 Compliance and Ethical Considerations

As technologies continue to advance and integrate into various sectors, addressing ethical considerations and ensuring legal compliance becomes increasingly imperative. All three models, ChatGPT, PaLM2, and Gemini, face similar challenges and considerations associated with LLMs, including:

- **Bias Mitigation:** Mitigating bias in systems is essential for ensuring fairness and representation. This involves balanced data collection and algorithmic adjustments to minimize biases. Continuous monitoring and bias testing are crucial to identify and address emerging biases, particularly intersectional biases and their causal interactions (McIntosh et al., 2023).

- **Data Security:** data security encompasses ensuring data confidentiality, adhering to consent norms, and safeguarding against vulnerabilities like membership inference attacks. Compliance

with legal standards such as GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act) is essential, requiring purpose limitation and data minimization. Robust encryption, access control, and continuous security assessments are critical for protecting user privacy (McIntosh et al., 2023).

- **AI Ethics:** The field of ethics focuses on fairness, accountability, and societal impact, addressing ethical challenges posed by AI's complexity and potential misalignment with human values. Ethical governance frameworks, multidisciplinary collaborations, and technological solutions are necessary. Ensuring traceability, auditability, and transparency throughout the model development lifecycle is crucial, but adoption of these initiatives remains uneven, emphasizing the need for consistent ethical practices (McIntosh et al., 2023).

- **Privacy Preservation:** Privacy preservation aims to maintain data confidentiality and integrity, employing strategies like anonymization and federated learning to minimize direct data exposure. Challenges such as achieving true anonymity against correlation attacks persist. Compliance with privacy laws and secure data handling practices are crucial, highlighting the continuous need for robust privacy preservation mechanisms (McIntosh et al., 2023).

2.11 Prompt Engineering

To communicate effectively with the mentioned LLMs (Gemini, PaLM2, and ChatGPT), prompt engineering was essential in crafting contextually appropriate inputs or questions to guide their responses.

In artificial intelligence, a prompt serves as an essential communication tool between humans and systems, guiding the model's response or action based on provided instructions or statements. It provides context, information, and direction to help models generate meaningful and accurate outputs. Prompts vary widely, ranging from simple keywords or questions to detailed descriptions or instructions tailored to the specific task at hand (Google Cloud, 2024b).

Prompt engineering focuses on crafting contextually appropriate inputs or questions to optimize the model's responses. This process considers the structure, content, and tone of the prompt to ensure clarity, relevance, and specificity in interactions. Effective prompt engineering is crucial for enhancing the quality of outputs from language models, facilitating accurate and engaging interactions across various applications (Bozkurt and Sharma, 2023).

The quality of outputs generated by models is highly dependent on the quality of prompts provided, highlighting the importance of strategic prompt design in achieving desired performance (Gong et al.,

2023).

2.11.1 Benefits of prompt engineering

To unlock the full potential of conversational generative AI, establishing clear and purposeful conversations becomes crucial, and this is precisely where prompt engineering assumes a pivotal role. Prompt engineering offers numerous benefits in enhancing the performance and usability of models (Google Cloud, 2024b):

- **Improved model performance:** Well-crafted prompts significantly enhance the quality and accuracy of model responses. By providing specific and relevant information, prompts help the model better understand user intent, leading to more precise and informative outputs.
- **Reduced Bias and Harmful Responses:** Prompt engineering mitigates bias and reduces the likelihood of generating harmful or nonsensical responses. Carefully crafted prompts guide the model to focus on specific task aspects, minimizing inappropriate or offensive outputs.
- **Increased Control and Predictability:** Prompt engineering provides greater control over the model's behavior. Designing prompts aligned with desired outcomes influences responses, ensuring consistency and predictability in outputs.
- **Enhanced User Experience:** Clear and concise prompts improve user experience by facilitating effective communication with the model. Users are guided on how to interact, leading to more satisfactory and intuitive interactions.

2.11.2 Best practices for prompt engineering

To write good prompts I consider the following strategies (Chen et al., 2023):

- **Start with a clear objective:** Write prompts that are straightforward and unambiguous. Avoid unnecessary complexity or vagueness, as it may lead to confusing or irrelevant responses.
- **Understand the model's capabilities:** Familiarize yourself with the strengths, limitations, and optimal query types of the language model.
- **Provide sufficient context:** Incorporate relevant context and examples to help the model better understand the task at hand.

- **Specify the format:** If you require a particular format or structure in the response, clearly indicate it in the prompt.
- **Test and iterate:** Experiment with different prompt variations and evaluate model responses to refine and improve prompt effectiveness.

By following these best practices, we can leverage prompt engineering to enhance the performance, reliability, and user experience of models.

2.12 Firebase Realtime Database

Firebase offers two cloud-based, client-accessible database solutions that support real-time data synchronization (Firebase, 2024):

- **Cloud Firestore:** Designed for mobile app development, Cloud Firestore enhances the Realtime Database with a more intuitive data model, faster and more advanced queries, and improved scalability.
- **Realtime Database:** Firebase's original database solution provides efficient, low-latency synchronization suitable for applications requiring real-time updates across clients.

Both options include client-centric SDKs that eliminate the need for server deployment, support real-time updates, and offer both a free tier and a pay-as-you-go pricing model (Firebase, 2024). In the broader context of Firebase solutions, while Cloud Firestore offers a newer model with more advanced querying capabilities, the Realtime Database remains our preferred choice due to its proven efficiency and low-latency operations for real-time applications.

The Firebase Realtime Database, developed by Google, is a cloud-hosted NoSQL database renowned for its real-time data storage and synchronization capabilities among users. It operates in JavaScript Object Notation (JSON) format, facilitating automatic synchronization across all devices and ensuring users have access to the latest data. Data within Firebase is organized using a JSON tree structure, where each node is uniquely identified by a string key and can store various JSON data types such as string, number, object, array, boolean, or null (Madaminov and Allaberganova, 2023).

Firebase Realtime Database differs from traditional Structured Query Language (SQL) databases by adopting JSON documents instead of tables. This approach enhances scalability and facilitates distributed data management across servers, ensuring high availability (Ohyver et al., 2019).

Key features of Firebase Realtime Database include (Firebase, 2024):

- **Real-time Updates:** Utilizes data synchronization instead of traditional HTTP requests, enabling real-time updates across all connected devices within milliseconds.
- **Offline Capabilities:** The Firebase Realtime Database SDK persists data locally on devices, ensuring apps remain responsive even without internet connectivity. Once online, devices sync seamlessly with the latest server data.
- **Direct Access from Client Devices:** The database can be accessed directly from mobile devices or web browsers without an application server. Security and data validation are enforced through Firebase Realtime Database Security Rules.
- **Scalability:** Supports data distribution across multiple databases under the Blaze pricing plan, facilitating large-scale data management needs.

2.12.0.1 Firebase Authentication

Firebase Authentication provides versatile back-end services, user-friendly SDKs, and ready-made UI libraries for user authentication in apps, supporting a range of methods including email/password, phone numbers, and federated identities like Google, Twitter, Facebook and GitHub. It can also integrate with existing account systems, providing flexibility in user management. We implemented Google Sign-In as part of our sign-up and sign-in mechanisms. This integration allows users to conveniently use their Google credentials to access our platform, simplifying the login process.

Upon successful authentication, three key things happen:

- **User Information Retrieval:** Information about the user is returned to the device through callbacks, allowing for a personalized app experience. This includes a unique ID that remains consistent across all providers and never changes for the authenticated user. This ID is crucial for identifying users and determining their access levels within the backend.
- **Session Management:** Firebase manages user sessions to ensure that users remain logged in even after a browser or application restarts, supporting a seamless user experience across Android, iOS, and web platforms.
- **Security and Access Control:** The system ensures that user data is securely handled, with Firebase Realtime Database Security Rules enabling expressive, rule-based access controls.

In our recruitment platform, a part of user authentication is managed through Firebase Authentication, as illustrated in the figure below. This system provides a user-friendly interface for managing user

credentials and authentication methods. The screenshot shows the Firebase Authentication dashboard, where user identities are managed:

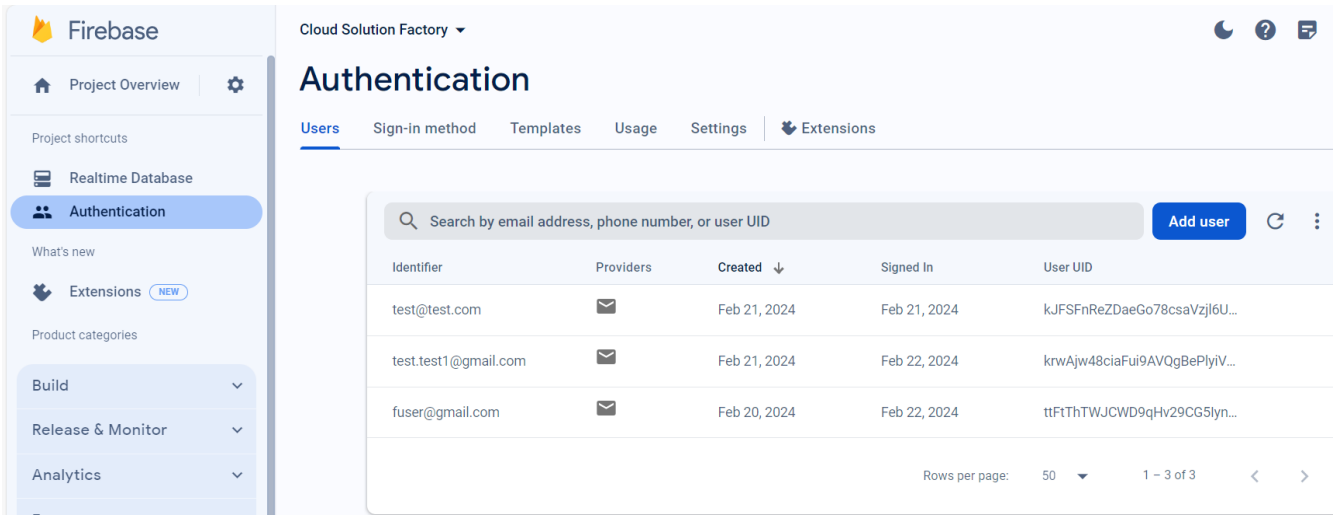


Figure 5: Firebase Authentication dashboard.

- **Identifier:** Displays the email address or other identifiers used for signing in.
- **Providers:** Indicates the method used for authentication, such as email/password or Google.
- **Created:** The date on which the user account was created.
- **Signed In:** The most recent sign-in date.
- **User UID:** A unique identifier assigned by Firebase to each user.

This setup ensures that each user's login information is securely managed and integrated seamlessly with our platform's backend services.

Chapter 3

Review of Artificial Intelligence Methods for Recruitment Processes

Companies are increasingly turning to to enhance their recruitment processes. The application of in recruitment encompasses deploying chatbots, automated resume assessments, personality profile creation, and conducting virtual interviews. These applications aim to boost the efficiency and effectiveness of recruitment processes (Tullier, 2021).

In this section, we will examine various artificial intelligence methods used in recruitment processes. We'll focus on how technologies are revolutionizing traditional hiring practices, improving efficiency, and bringing new methods to candidate evaluation and selection. We will explore the integration of tools in different stages of recruitment, from resume screening to interview analysis, and discuss the implications of these advancements for both employers and job seekers. This exploration aims to offer a thorough understanding of the changing role of in recruitment and its impact on the future of talent acquisition.

3.1 Hirevue

HireVue, founded in 2004, stands at the forefront of cutting-edge AI-powered recruiting technology. HireVue's approach represents a paradigm shift from traditional hiring practices, focusing on candidates' skills and potential rather than just their resumes (HireVue, 2024).

HireVue employs several AI-driven mechanisms to enhance the recruitment process:

- **Automated Screening:** HireVue's algorithms efficiently analyze large volumes of applications to identify the most promising candidates based on predefined criteria. This automation significantly reduces the time recruiters spend on initial resume reviews, allowing them to focus on more critical aspects of the hiring process (Jr. and Giffen, 2019).

- **Video Interview Analysis:** Candidates can complete video interviews at their convenience. Hire-

Vue's processes these interviews by analyzing both video and audio inputs (van Esch and Black, 2019). This involves:

  - **Verbal and Non-Verbal Cue Analysis:** analyzes tone, pitch, body language, and facial expressions during the video interviews. This analysis helps assess communication skills, confidence, and other relevant attributes (Trziszka, 2023).
  - **Content Evaluation:** The also evaluates the content of candidates' responses for keywords, coherence, and relevance to the job role. This combined analysis streamlines the scheduling and evaluation process, making it particularly beneficial for candidates currently employed or in different time zones (Trziszka, 2023).

- **Gamified Assessments:** HireVue incorporates AI-powered games to assess cognitive abilities, problem-solving skills, and behavioral traits. These gamified assessments provide insights into a candidate's potential job performance and cultural fit (Jr. and Giffen, 2019).

- **Algorithmic Matching with Find My Fit:** This feature uses to match candidates to suitable roles based on their skills, interests, and background. The compares candidates' profiles with job requirements, recommending the best fits and expanding the diversity and reach of the talent pool (HireVue, 2024).

- **Structured Interview Process:** HireVue's ensures a structured interview format with standardized questions for all candidates, which helps eliminate variability that can arise from unstructured, subjective interviews (HireVue, 2024).

- **Algorithmic Fairness and Bias Reduction:** HireVue's is designed to be objective, focusing on measurable data points rather than subjective judgments. Regular audits and updates ensure that the does not inadvertently introduce biases. Additionally, HireVue can anonymize candidate information during the initial screening stages, ensuring decisions are based solely on performance and responses, rather than potentially biased information such as name, gender, or ethnicity (HireVue, 2024).

HireVue has been adopted by over 700 companies, including notable names such as Unilever, Vodafone, PwC, and Oracle. The implementation of HireVue's technology has led to significant improvements in the recruitment process, including a reduction in recruitment time by 90% and an increase in employment diversity by 16%. These benefits highlight HireVue's ability to enhance recruitment efficiency, improve the

quality of hires, provide convenience for candidates, and reduce biases in the hiring process (Trziszka, 2023).

3.2 Pymetrics

Pymetrics, founded in 2013, is a startup that leverages ML and neuroscience-based assessments to enhance the recruitment process. The platform uses interactive games to evaluate candidates' cognitive, social, and emotional skills, aiming to create a more data-driven, unbiased, and effective hiring process (Pymetrics, 2024).

How Pymetrics Uses in Recruitment:

- **Neuroscience-Based Game Assessments:** Pymetrics uses 12 interactive games rooted in neuroscience and peer-reviewed psychological studies. These games measure a range of attributes including memory, attention, risk-taking, and fairness. The games collect data on various cognitive and emotional traits, providing a comprehensive profile of each candidate (Pymetrics, 2024).
- **Machine Learning Analysis:** Pymetrics develops ML algorithms tailored to each company's needs. These algorithms analyze the game data to identify traits and patterns associated with successful employees in specific roles. The algorithms are rigorously tested for bias to ensure they do not inadvertently favor any group, promoting fairness and ethical recruitment practices (Wilson et al., 2021).
- **Candidate-Job Matching:** AI-driven algorithms match candidates to suitable job roles based on their game performance and the success profiles of current employees. This increases the precision of matching candidates to roles where they are likely to succeed, enhancing overall job satisfaction and performance (Wilson et al., 2021).
- **Interview Process Optimization:** Based on the data from the game assessments, Pymetrics provides personalized interview questions. This helps standardize the evaluation process and reduce interviewer bias by focusing on relevant skills and behaviors. The platform allows candidates to practice responses and offers privacy options, such as hiding their camera. This feature helps candidates feel more comfortable and prepared during the interview process (Pymetrics, 2024).

Companies using Pymetrics have reported an 18% increase in employee tenure, suggesting better long-term job fit. The platform has helped reduce the time to hire by 59%, making the recruitment process

faster and more efficient. There has been a reported 28% increase in annual sales, indicating that better candidate-job matching can positively affect business performance. Pymetrics has significantly improved diversity in hiring, with a 62% increase in female representation and a doubling of offers extended to minority candidates. Pymetrics' innovative approach to recruitment has been adopted by several major companies, including Unilever and LinkedIn. These organizations have benefited from the platform's ability to enhance recruitment efficiency, improve candidate-job fit, and promote a more diverse and inclusive workforce (Pymetrics, 2024).

3.3 Eightfold

Founded in 2016, Eightfold leverages deep learning and a robust global talent dataset to empower organizations in acquiring, managing, and developing diverse talent effectively. Eightfold's Talent Intelligence Platform is a transformative AI-driven tool that revolutionizes talent management, aiming to provide equitable career opportunities globally (Hoole et al., 2023).

How Eightfold Utilizes in Recruitment and Talent Management:

- **AI-Powered Talent Assessment:** Eightfold employs deep learning AI, fueled by a global talent dataset encompassing over 1.4 million unique skills. algorithms analyze and correlate skills data to precisely match candidates with job roles based on their qualifications and potential (Hoole et al., 2023).

- **Automated Recruitment Processes:** The platform automates sourcing, screening, and scheduling of candidates, significantly reducing manual workload for recruiters. algorithms analyze candidate profiles from various sources, including resumes, online profiles, and public/private databases, to identify qualified applicants efficiently (Eightfold AI, 2024a).

- **Bias Reduction:** Eightfold uses AI-driven masked profiles to anonymize personal details like gender and ethnicity during candidate evaluation. This ensures fair and unbiased assessment based solely on qualifications and potential (Eightfold AI, 2024a).

- **Personalized Career Recommendations:** Eightfold provides candidates with personalized job recommendations aligned with their skills and career aspirations. This enhances candidate engagement and satisfaction while enabling recruiters to focus strategically on building meaningful connections with candidates rather than administrative tasks (Hoole et al., 2023).

- **Comprehensive Talent Management:** Eightfold's Career Hub empowers employees to explore growth opportunities within the organization through personalized development paths, mentor matching, and skill-based learning programs. The platform facilitates succession planning by identifying potential successors for critical roles based on skill readiness and organizational needs. Real-time insights into team potential and individual performance enable managers to implement proactive talent management strategies aligned with business objectives (Eightfold AI, 2024b).

Eightfold's Talent Intelligence Platform has played a pivotal role in transforming recruitment and talent management practices across various organizations. Operating in over 145 countries and supporting 19 languages, it adheres to stringent privacy and security standards. For instance, its implementation at Vodafone led to a remarkable 144% surge in female applicants within the first three months, underscoring its pivotal role in fostering diversity initiatives. At Dexcom, Eightfold significantly reduced interview scheduling time by 33%, showcasing its efficiency in optimizing recruitment processes. Moreover, Tata Communications reported an impressive 81% increase in internal hires, demonstrating Eightfold's capability to enhance internal talent mobility and facilitate career advancement opportunities within organizations. These outcomes highlight Eightfold's ability to leverage effectively, not only in improving operational efficiencies but also in promoting inclusive hiring practices and nurturing internal talent development (Hoole et al., 2023).

3.4 HireEZ

Formerly known as Hiretual, HireEZ specializes in AI-powered solutions tailored to streamline various aspects of the recruitment lifecycle.

- **AI-Powered Sourcing:** HireEZ taps into a vast ecosystem of over 40 platforms, aggregating data from more than 700 million candidate profiles and 100 million job descriptions. This extensive reach allows recruiters to identify and source potential candidates efficiently. Central to its sourcing capabilities is a robust knowledge graph enriched with conversational utterances and linguistic logic. This graph enhances candidate screening and shortlisting by mapping relevant attributes to job positions and candidate profiles (Veluchamy et al., 2021).

- **AI-Driven Engagement Solutions:** HireEZ employs strategies to improve candidate engagement. Through personalized communication and automated follow-ups, recruiters can build effective candidate pipelines, fostering a responsive and engaged talent pool. HireEZ's AI-driven

chatbot, Wendy, plays a pivotal role in engaging candidates throughout the recruitment process. Wendy conducts personalized conversations, assesses candidate interest and qualifications, and schedules interviews. For instance, a Fortune 100 manufacturing firm utilized Wendy to engage hundreds of candidates, resulting in successful interview scheduling for 325 qualified applicants (Veluchamy et al., 2021).

Implementing HireEZ has shown tangible benefits such as a 10% reduction in time-to-hire, a 50% decrease in sourcing time, and a 30% increase in response rates for organizations like Chenega Corporation (Veluchamy et al., 2021). The integration of not only enhances efficiency but also scalability, enabling recruiters to manage large volumes of data and interactions seamlessly.

3.5 Textio

Founded in 2014, Textio offers augmented writing solutions that utilize to enhance the efficiency and effectiveness of the hiring process through various tools and capabilities:

- **Textio Flow:** This tool enables recruiters to create job advertisements quickly by suggesting and completing content based on AI-driven insights. It simplifies the content creation process, ensuring that job postings are optimized for clarity, engagement, and inclusivity from the outset (Veluchamy et al., 2021).
- **Hiring Score:** Textio provides predictive analytics through Hiring Score, which evaluates the language used in job postings. It offers insights into how effective a job posting is likely to be in attracting top-tier candidates. Recruiters receive guidance on optimizing their recruitment strategies to enhance candidate attraction and engagement (van Esch and Black, 2019).
- **Bias Reduction:** Textio helps recruiters identify and eliminate hidden gender bias in job postings by suggesting alternative language that promotes diversity and inclusivity. This feature expands the applicant pool by appealing to a broader range of candidates (Veluchamy et al., 2021).
- **Central Library:** Serving as a repository, Textio's Central Library ensures consistency in hiring documents and key phrases used across job postings. This feature helps maintain alignment with a company's culture and brand voice, enhancing overall communication effectiveness (Veluchamy et al., 2021).

Textio's AI-powered solutions have been instrumental in transforming the recruitment processes of leading organizations such as Cisco, McDonald's, and Johnson & Johnson. By integrating into their initial recruitment stages, these companies have realized significant improvements. For example, Procter & Gamble reported a notable 30% increase in the number of qualified applicants after implementing Textio (Veluchamy et al., 2021). This outcome underscores Textio's effectiveness in optimizing job postings to attract a higher caliber of candidates.

3.6 Paradox

Founded in 2016, Paradox Technology is an innovative recruitment platform that leverages to automate and streamline various aspects of the hiring process, aiming to enhance efficiency and effectiveness.

How Paradox Technology utilizes AI (Paradox AI, 2024):

- **Automated Recruitment Tasks:**
  - Sourcing: algorithms search and identify potential candidates across platforms based on job descriptions and profiles.
  - Screening: AI-driven tools assess candidate qualifications against job requirements, customizable to specific roles.
  - Scheduling: Automates interview scheduling, integrating with calendar systems for real-time updates and candidate self-scheduling options.

  .

- **AI-Driven Recruitment Assistant (Olivia):** Engages candidates in over 30 languages through interactive dialogs. Collects comprehensive candidate information, including introduction videos.
- **Comprehensive Insights:** The platform offers robust reporting capabilities with dashboards providing valuable insights into candidate satisfaction, response rates, and other key recruitment metrics. These insights empower data-driven decision-making to continually refine recruitment strategies.

Paradox Technology has achieved impressive results: it maintains a remarkable 99% satisfaction rate by prioritizing candidate experience throughout the recruitment journey. The platform significantly reduces job advertising costs by up to 50% through AI-driven advertising strategies. Moreover, it improves hiring

outcomes by 30%, particularly for challenging roles (Paradox AI, 2024). The platform's capabilities have resulted in significant improvements in recruitment efficiency and candidate satisfaction rates.

3.7 Findem

Launched in 2020, Findem is an advanced recruitment platform that harnesses the power of to streamline the talent acquisition process. Findem integrates various AI-driven functionalities to enhance the accuracy and efficiency of hiring processes (Findem AI, 2024):

- **AI-Driven Search and Personalization:** Findem allows users to create personalized searches using various criteria, such as LinkedIn profiles or job descriptions. This search capability is powered by an AI-driven assistant, similar to ChatGPT, enriched with detailed data about people and companies. The platform builds diverse talent pools by identifying the most suitable candidates and crafting highly personalized and accurate email communications. This AI-enhanced personalization outperforms other solutions in terms of accuracy and effectiveness, significantly improving the recruitment workflow.

- **GenerativeAI and Conversational Analytics:** The integration of generativeAI and conversational analytics makes Findem's Assistant an essential part of every aspect of the talent search process. From profile outreach and assessment to analytical dashboards, the Assistant simplifies identifying qualified candidates who match specific hiring criteria. This intelligent automation significantly reduces the time spent screening applications, allowing recruiters to spend more time engaging with ideal talent for the organization.

- **Improved Candidate Selection and Search Refinement:** Findem's innovation extends beyond basic search functionality. The platform improves candidate selection by analyzing sets of individuals who meet specified criteria and suggesting additional adjustments to refine the search. For example, when searching for 'Java Developers in the Bay Area with Startup Experience,' the platform not only identifies suitable candidates but also offers options to enhance the search based on profile analysis of the candidates found. This iterative refinement process ensures that recruiters can find the best match for their needs.

- **Intelligent Automation and Analytical Insights:** Findem's Assistant leverages intelligent automation to manage and process large volumes of candidate data efficiently. The platform's analytical dashboards provide valuable insights into the recruitment process, helping recruiters make

data-driven decisions. By automating routine tasks and offering deep analytical insights, Findem enhances the overall effectiveness of recruitment teams.

- **Integration and Continuous Improvement:** Findem continuously learns and improves from its interactions, ensuring that its AI-driven processes become more accurate and effective over time. This continuous improvement makes Findem a valuable asset for recruitment teams aiming to stay ahead in the competitive talent acquisition landscape.

The platform's capabilities streamline the talent search and engagement process, making it more efficient and accurate. This efficiency leads to a significant reduction in time spent on manual tasks, enabling recruiters to focus on strategic engagement and personalized interactions with candidates (Findem AI, 2024). In summary, Findem's Assistant establishes itself as a highly effective recruitment tool by combining the flexibility of a chatbot with the accuracy of a search system enriched with industry-specific data.

3.8 Humanly.io

Launched in 2019 by an American company, Humanly.io is an innovative chatbot specifically designed for the HR sector. This platform integrates with business operations, addressing the challenges of a digitalized society. Its primary role is to revolutionize the recruitment process, offering vital support to candidates, particularly those with less experience or attending their first job interviews.

How AI is employed at Humanly.io (Cimpeanu, 2023):

- **AI-Driven Interaction and Candidate Support:** Humanly.io leverages to facilitate seamless interactions between HR specialists and candidates. The chatbot engages with candidates, providing essential information and guidance throughout the recruitment process. This is particularly beneficial for young candidates or those new to the job market, as it prepares them effectively for their interviews and roles.

- **Global Talent Acquisition:** The platform enables recruiters to identify potential candidates globally, extending talent acquisition beyond geographical limits. This broad reach allows companies to tap into a diverse talent pool, ensuring they find the best fit for their roles regardless of location.

- **Rapid Reference Checks:** One of Humanly.io's standout features is its ability to conduct rapid reference checks for various positions. By automating this process, the platform enhances the

efficiency and reliability of candidate verification, ensuring that only the most qualified candidates move forward in the hiring process.

- **Automated Screening and Data Management:** Humanly.io excels in managing and classifying vast volumes of candidate data. Its AI-driven automatic screening feature checks and analyzes all registered candidates against the requirements of available positions. This automation allows the platform to suggest specific questions or tasks for candidates, enabling a thorough and effective evaluation tailored to the company's needs.

- **Enhanced Recruitment Efficiency:** By automating many routine tasks, Humanly.io allows HRprofessionals to focus on more strategic aspects of recruitment. The platform's continuously learns and improves, making the recruitment process faster and more efficient over time. This leads to a significant increase in the number of candidates for available positions and enhances overall employee satisfaction.

Companies that implement Humanly.io report increased candidate numbers for job openings and higher employee satisfaction rates. The automation of routine tasks also frees up time for employees to focus on personal and professional development (Cimpeanu, 2023).

3.9 HiredScore

Founded in 2013, HiredScore is a recruitment platform that leverages artificial intelligence to enhance and streamline the hiring process. Here's a detailed look at how it uses and how the platform functions (HiredScore, 2024):

- **Automated Screening and Ranking:** HiredScore's begins by analyzing large volumes of candidate data, including resumes, professional histories, and responses to screening questions. The platform uses advanced ML algorithms to process this information, learning from past hiring decisions and employee success patterns to continuously refine its analysis. By doing so, it ranks candidates based on how well they match the job requirements and preferences. This ranking process helps recruiters quickly identify the most suitable candidates from a large pool, significantly reducing the time spent on initial screening.

- **Intelligent Matching:** Beyond basic qualifications and experience, HiredScore's incorporates more nuanced factors into its matching process. It correlates candidates to open positions by

considering elements such as a candidate's potential for growth and their likelihood of accepting an offer if made. This intelligent matching ensures that recruiters not only find candidates who are technically qualified but also those who are likely to fit well within the company and its culture, leading to better long-term hires.

- **Predictive Analytics:** HiredScore uses predictive analytics to forecast a candidate's future performance and their probability of staying with the company long-term. By employing ML techniques, the platform analyzes patterns and outcomes from previous hires to make these predictions. This allows recruiters to make more informed decisions, selecting candidates who are not only a good fit for the role but also likely to contribute positively to the company over time.

These features collectively make HiredScore a robust tool for recruiters, enabling them to quickly and accurately identify the best candidates, thus improving the overall efficiency and effectiveness of the hiring process.

3.10 Comparing all platforms

This concluding section synthesizes the insights gleaned from each technology, reflecting on how they enhance the efficiency, fairness, and effectiveness of recruitment practices.

Each technology brings unique strengths to the recruitment process:

- **HireVue:** Offers automated resume screening, video interview analysis, and gamified assessments for skill evaluation, aiming for objective and efficient candidate assessment.
- **Pymetrics:** Uses neuroscience-based games to assess cognitive and emotional skills, improving candidate-job fit and reducing bias in the hiring process.
- **Eightfold:** Utilizes deep learning and extensive data to match candidates to roles based on skills and potential, promoting inclusive talent management.
- **HireEZ:** Uses for candidate sourcing and engagement, with a focus on creating talent pipelines and automating personalized communications.
- **Textio:** Focuses on augmented writing solutions to optimize job postings with inclusive language and predictive analytics to enhance candidate attraction.

- **Paradox:** Automates recruitment tasks like sourcing and scheduling, emphasizing candidate experience customization and detailed insight reporting.
- **Findem:** Integrates AI-driven search and personalized outreach to build diverse talent pools and recommend roles based on specific skills.
- **Humanly.io:** Uses chatbots for candidate interaction, offering global support in talent acquisition and rapid reference checks.
- **HiredScore:** Focuses on automation for initial resume screening and candidate ranking using ML algorithms, aiming for efficiency in identifying suitable profiles.

None of the platforms mentioned above utilize genAI to generate customized technical questionnaires like AssessBot, which distinguishes this platform by offering a unique and highly personalized approach to technical assessment of candidates. Additionally, AssessBot will offer comprehensive dashboards for tracking candidate performance on these assessments and may incorporate Curriculum Vitae (CVs) screening capabilities, distinguishing itself with a multifaceted approach to candidate evaluation.

It is evident that plays a crucial role in the recruitment process. The integration of these technologies in recruitment processes has led to significant improvements in efficiency, the quality of hires, and candidate experience. However, it is crucial to navigate the use of in recruiting responsibly, ensuring fairness and maintaining the human touch. While these technologies offer tremendous benefits, they also present challenges that need to be addressed, such as ensuring privacy, avoiding algorithmic biases, and maintaining transparency.

Chapter 4

Implementation

In this chapter, I delve into the practical aspects of the platform, focusing on the implementation process. It begins with familiarization with VertexAI, the platform used to work with the PaLM2 and Gemini models. Initially, the focus was on training the PaLM2 model since the Gemini model was not available at that time. After the training, a comparative analysis was conducted between PaLM2 and GPT-3.5. Once the Gemini model became available, we transitioned to using it and performed a feasibility analysis to evaluate its results.

Next, I present the architectural solution, including the domain model, application workflow, database structure, and platform design, accompanied by screenshots of the entire application. Finally, I conclude with a cost estimation and the deployment process.

4.1 Familiarization with VertexAI

First, let's provide a brief contextualization of VertexAI and how it was utilized. Vertex AI, as mencioned before, is an artificial intelligence platform developed by Google, designed to facilitate the development, training, and deployment of ML models at scale. In this context, two specific models were employed: Text-Bison-32k (PaLM2) and Gemini Pro 1.5.

To ensure the effectiveness of these models, specific parameters were defined. They include:

- **Temperature:** This parameter controls token selection randomness, ranging from 0 to 1. Lower temperatures suit prompts expecting precise responses, while higher ones result in more diverse outcomes but more randomness. A temperature of 0 selects the highest probability token results in more deterministic responses. In our task, we set it to 0.2 for a balance between diversity and coherence. With higher temperature values, we observed the introduction of more random questions that often lacked coherence with the context.

- **Token limit:** This parameter determines the maximum number of tokens that can be generated in the response, with a range from 1 to 8192. Given that a token is approximately four characters long and corresponds to about 60-80 words per 100 tokens, the chosen limit was 8192 tokens. Since our goal was to construct a questionnaire with 15 questions, each accompanied by 4 response options and one correct answer, we needed to set the value higher to ensure that the responses were long enough to present the entire quiz.

- **Top-k:** This parameter changes how the model selects tokens for output. A top-k of 1 means the selected token is the most probable among all tokens in the model's vocabulary (also called greedy decoding), while a top-k of 3 means that the next token is selected from among the 3 most probable tokens (using temperature). The top-k parameter was set to 4, ranging from 1 to 40. This value was chosen to balance token selection between the highest probability and a slight diversification in responses. This allows for some flexibility in response generation while ensuring that the most probable options are prioritized.

- **Top-p:** This parameter alters how the model selects tokens for output, determining the threshold at which tokens are considered during generation. Tokens are selected from most probable to least until the cumulative probability equals the top-P value. For instance, if tokens A, B, and C have probabilities of 0.3, 0.2, and 0.1, respectively, and the top-P value is 0.5, the model will select either A or B as the next token, using temperature to make the final selection. The top-P parameter ranges from 0 to 1, with 0 indicating that only the most probable token is chosen and 1 meaning that all tokens with cumulative probabilities less than or equal to 1 are considered. In this scenario, a value of 0.5 was selected to strike a balance between generating diverse responses and maintaining coherence with the prompt.

Understanding these parameters is fundamental to optimizing the models' performance and ensuring precise and consistent results. Before delving into the distinction between freeform and structured prompts, it's important to understand what constitutes a prompt. A prompt is text input used to generate a model response. Prompts can include a preamble, questions, suggestions, instructions, or examples. With this understanding, let's now explore the differences between freeform and structured prompts, as the model supports both formats. Freeform prompts allow for a more flexible and natural interaction with the model. In this mode, users are not constrained by specific formats or templates, enabling them to input queries, commands, or prompts in a conversational manner. Freeform prompts are particularly suitable for scenarios where users want to engage with the model in a less rigid and more intuitive manner,

such as asking open-ended questions or generating creative text outputs. On the other hand, structured prompts follow a predefined format and include examples to guide the model in understanding the desired output format. Examples are often used to guide the model in understanding the desired output format. These examples serve as reference points for the model, helping it generate responses that adhere to the specified structure. Structured prompts are beneficial in situations where specific information must be provided in an ordered manner, ensuring that the model understands and generates outputs that align with the desired structure.

We initially started with freeform prompts, but quickly realized the need to incorporate examples of questions to improve the model's performance. As a result, model training was conducted using structured prompts.

4.2 Model Training

This section delves into the intricate process of training the model, employing prompt engineering techniques to optimize its performance. The goal was to achieve a refined output, ensuring diverse and pertinent questions tailored to specific requirements. The training process involved utilizing the Text-Bison-32k model, as the Gemini model was not yet available at that time. Given our specific requirements, the structured prompt format was selected for training, primarily due to its compatibility with examples. By structuring prompts with specific examples, we provided the model with clear guidance to enhance its comprehension of contextual nuances and task intricacies. These examples were sourced from GeekforGeeks, a reputable platform known for its comprehensive technical content.

4.2.1 PaLM2 - Text-Bison-32k (latest)

First, let's provide a brief overview of how the VertexAI console works for generating prompts. In the image below, we can see on the right side the section where we choose the parameters discussed in the previous section. Then, we have several fields to fill out, such as context, examples, and test. Next, we'll provide a description for each of these fields.

- **Context:** Instructs how the model should respond. For example, "Explain this code", specifying words the model can or cannot use, topics to focus on or avoid, or response format. Context applies each time you send a request to the model
- **Input:** This is where you make a request to the model. When you submit a request, the context,

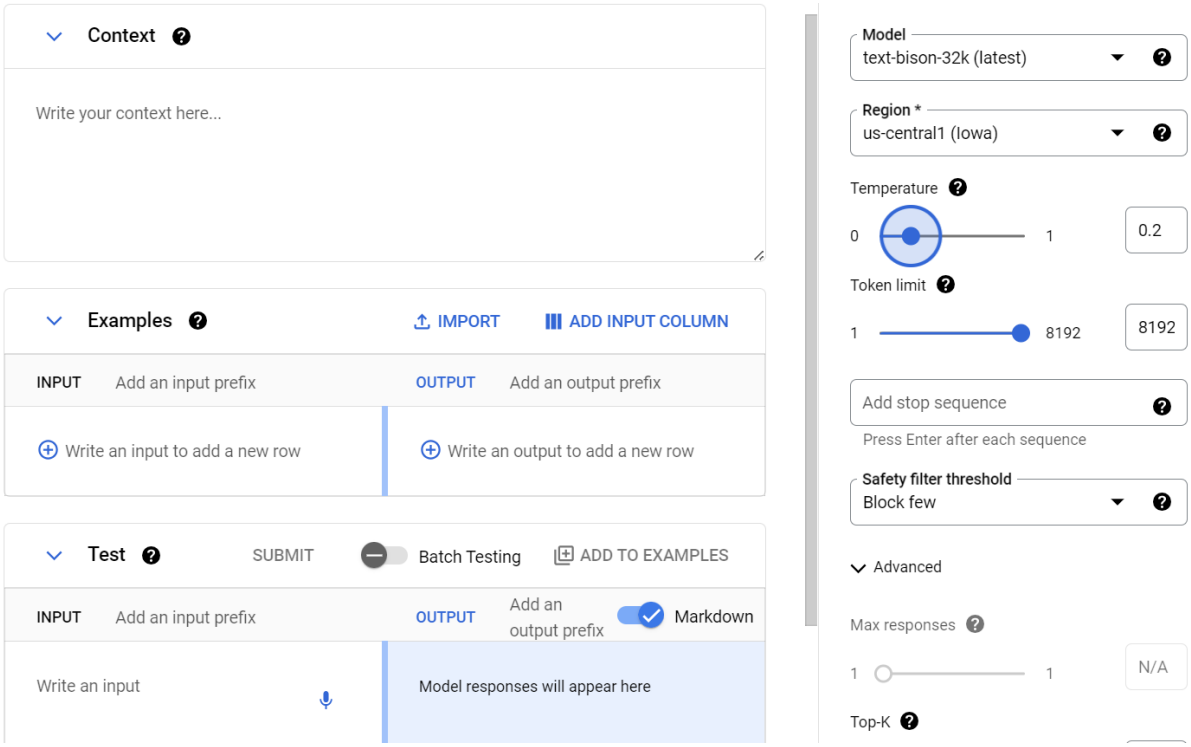


Figure 6: VertexAI console.

examples, and input fields are all sent together.

- **Output:** This is your model's response, which takes into account all of the prompt fields (context, examples, input).

To assess the model's ability to generate quality multiple-choice questionnaires, we focused on six key technical areas (DevOps, Java, Cybersecurity, iOS, Python, and React) across three proficiency levels (Junior, Mid/Intermediate, and Senior). However, we encountered an early challenge in the model's inability to distinguish between different seniority levels, resulting in equally challenging or easy questions being posed to both junior and senior levels. To address this issue, we prioritized training the model to differentiate between junior and senior levels before introducing the intermediate level. This adjustment aimed to refine the model's understanding of proficiency levels and ensure the appropriate difficulty of questions for each level.

Next, we will present the structure of the context, examples and test fields used, discussing how the prompts evolved based on the issues encountered in the questionnaires.

To address the difficulty in distinguishing proficiency levels, we adjusted the prompts to include more nuanced examples and contextually relevant information. This evolution aimed to guide the model towards generating questions that accurately reflect the intended proficiency levels and align with the complexity

of the technical requirements.

Additionally, we iteratively refined the prompts based on feedback from manual evaluations. By analyzing the effectiveness of the generated questionnaires and identifying opportunities for improvement, we continuously adapted the prompts to enhance the model's proficiency level differentiation and question quality.

Let's analyze the evolution process of the model's fields, focusing on a specific case of a senior DevOps developer.

Initially, the context used was "Imagine that you are recruiting DevOps senior developers for a job and you have to make a quizz to understand their knowlege. Proficiency in version control systems such as Git, experience with CI/CD (Continuous Integration and Continuous Delivery) tools like Jenkins and Travis CI, and skills in infrastructure as code using tools like Terraform and Ansible are essential knowledge areas. Expertise in containerization and orchestration with Docker and Kubernetes is crucial, as well as the ability to implement and manage monitoring and logging systems. Moreover, solid knowledge in cloud computing, with platforms such as AWS, Azure, and Google Cloud, is necessary. Security concerns for applications and infrastructure should always be a focus, along with skills in scripting and automation, preferably in languages like Bash or Python. A good understanding of automated testing, networks, and communications is also vital." The context was crafted to simulate a real-world scenario where a hiring manager needs to assess the expertise of senior DevOps developers. Its purpose was to provide a comprehensive overview of the essential knowledge areas and skills required for a senior DevOps role. By giving the model concrete examples of areas to cover the context aimed to guide the model in generating questions covering a wide range of relevant topics within the DevOps domain.

The initial input began with "Generate a multiple choice test with 10 questions to assess a DevOps senior developer. Give me the correct answer for each question." This input set the task of creating a quiz aimed at evaluating the knowledge of senior DevOps developers, with the requirement for correct answers to each question.

However, several issues were encountered during this process:

- **Repetitive Questions or Answers:** The model often generated questions with repetitive content, where either the questions themselves were repeated or the answer options remained consistent across multiple questions.

- **Frequent Use of "All of the Above":** The phrase "all of the above" was frequently utilized as an answer option, appearing excessively in the generated quizzes. Moreover, when "all of the above" was present as an option, it often corresponded to the correct answer, leading to predictable

patterns in the quiz responses.

- **Uniform Question Structure:** Questions often had a similar structure, lacking diversity. For example, many questions began with the phrase "What is a ...?" leading to a monotonous pattern in the generated questions.
- **Difficulty Distinguishing Seniority Levels:** The model struggled to differentiate between different levels of seniority, resulting in questions of similar difficulty for both junior and senior levels.
- **Ambiguous Questions:** Some questions were ambiguous or lacked clarity.
- **The model takes time to learn:** The model displayed a need for further training, requiring additional data to enrich its dataset and broaden its comprehension of diverse question formats and topics.

To address these issues, efforts were made to vary the structure of the questions, introducing different formats and phrasings to provide a more diverse set of questions. By doing so, the aim was to make the quiz more engaging and challenging for the candidates, as well as to ensure a more comprehensive assessment of their knowledge across various aspects of DevOps.

Additionally, a pivotal refinement was made within the "examples" field. Initially, questions were inputted into the "examples - input" field, and the corresponding answers were provided in freeform text within the "examples - output" field. However, recognizing the need for a more structured approach to facilitate the model's understanding and output generation, a new method was adopted.

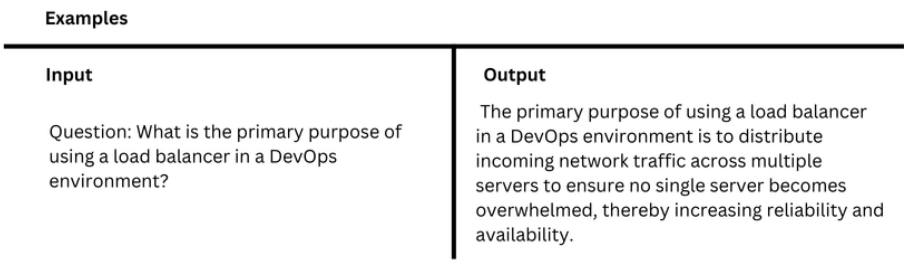


Figure 7: Examples fields before refinement.

In the "examples - input" field, questions were inputted alongside the respective four answer options. Notably, the format of the answer options was changed from freeform text to structured options. Subsequently, in the "examples - output" field, the correct answer for each question was specified in the format

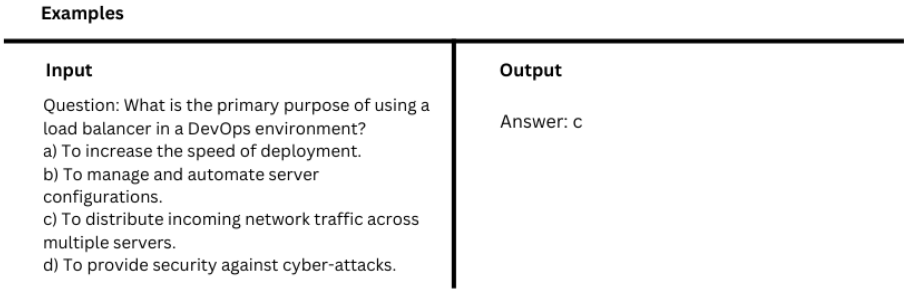


Figure 8: Examples fields after refinement.

"Answer: option". The output within the "examples" field now consistently provides the correct answer for each question in a clear and succinct manner.

This enhancement in handling input and output within the "examples" field significantly contributed to the overall enhancement of the model's performance and the quality of the generated quizzes. The input prompt was refined to "Generate a multiple-choice test with 10 high-difficulty questions to assess a senior DevOps developer. Each question should have a unique structure and avoid generic responses like 'all of the above'. The questions should include practical scenarios and multiple-choice questions with a single correct answer, focusing on advanced challenges and practical solutions in each mentioned area. Give me the correct answer for each question."

This specification directly addressed the issues we encountered previously. While the context remained the same, we adjusted the structure of the examples to directly address the problems identified with the initial input prompt, leading to better results in the quality and diversity of the generated quizzes.

In summary, through continuous adjustments the model has demonstrated significant improvements in generating questionnaires, which signify a positive evolution in the model's performance.

4.3 PaLM2 vs GPT 3.5

In this section, the goal is to express that despite the improvements observed, the questionnaires still fell short of the expected standard. As a result, the team decided to explore an alternative approach using ChatGPT. The reason for this shift was due to the mixed results obtained with PaLM2. While PaLM2 offered integration with Google Cloud and the potential for customization based on specific data, it struggled with generating repetitive and less diversified questions. Moreover, it exhibited inconsistency in tailoring questions to candidates' experience levels, occasionally presenting overly simplistic questions to senior candidates. These limitations were attributed to the model's narrower training data, affecting the quality

and relevance of the questions produced.

In contrast, ChatGPT presented several advantages, including the ability to handle more complex and diverse questions, adapting to different difficulty levels and contexts. Its training on vast datasets contributed to greater precision and relevance in the generated questions. However, leveraging ChatGPT also posed challenges, such as the need for supervision and adjustments to ensure questions and responses align with recruitment needs.

As the Gemini model became accessible, we transitioned to its utilization, capitalizing on its advanced capabilities and multimodal functionality. It was with the Gemini model that we conducted the feasibility analysis, which we will examine next.

The decision to transition from PaLM2 to ChatGPT and then to Gemini was driven by the need for more robust performance in generating high-quality questionnaire content that meets diverse candidate needs across different experience levels and contexts.

4.4 Feasibility Analysis

With the introduction of Gemini, we conducted performance tests to assess its capabilities. The results showed significant improvements, particularly in the model's ability to differentiate between difficulty levels accurately. As a result, we transitioned to a more comprehensive analysis of the outcomes, aiming to validate the system's effectiveness thoroughly. This analysis aims to validate the viability of the system and gather statistical evidence to support its effectiveness. To achieve this, an automation script was developed to generate questionnaires automatically, complemented by a comprehensive manual review process. The generated questionnaires were tailored to various technical roles (DevOps, Java, CyberSecurity, IOS, Python and Reach) and seniority levels (Junior, Mid/Intermediate and Senior) across different technology domains. The script's primary goal is to produce diverse sets of questions tailored to various technical roles and seniority levels, ensuring a broad and accurate assessment framework for technical evaluations. Each questionnaire comprised 15 questions, with a total of 10 tests generated for each technical role and seniority level.

At this point, it was necessary to think of a more general prompt. Previously, we had a specific prompt for each technology, but now, for the final system, it needs to be general. Below is the prompt we used:

```
Create a challenging multiple-choice test consisting of {NumberQuestions}
questions designed to evaluate a {Experience} {Role}'s proficiency at a
high difficulty level. Each question should have a distinctive format, and
```

```
the correct answers must be different from question to question and avoid
generic responses like "All of the above". Incorporate practical scenarios
and single-answer 4 answer multiple-choice questions. Do not incorporate
questions with code snippet. Additionally, provide the correct answer for
each question. Output the result using the following JSON format structure
replacing where the character # should be replaced by the questions, possible
choices and correct answers:

{
    "Questions": [
        {
            "Question": "#",
            "Answers": [
                "#"
            ],
            "CorrectAnswer": "#"
        }
    ]
}
```

The specification of a specific JSON format was necessary to ensure consistency and ease of processing for the generated test results. In the context provided, the JSON structure was defined as follows:

- Questions: This key encapsulates an array where each element represents an individual question.
- Question: Each question object within the array contains:
  - Question: The text of the question itself, represented by # which should be replaced with the actual question text.
  - Answers: An array holding the possible choices for that question, represented by # which should be replaced with the actual answer choices.
  - CorrectAnswer: Specifies which choice from the Answers array is the correct answer for that question, also represented by #.

In addition to the specified JSON format and context, the model also takes into account the job description provided by the recruiter to tailor the generated questionnaires. This ensures that the questions are aligned not only with the technical proficiency required for the role but also with the specific skills and competencies outlined in the job description.

Key Functionalities of the Script:

- **Questionnaire Generation:** Produces sets of questions (10 questionnaires for role-level with 15 questions each) tailored to various technical roles and seniority levels, across different technology domains.
- **Longest Correct Answer Check:** Automatically identifies if the correct answer is the longest among the options, helping to avoid predictable patterns in answers.
- **Unique Question Validation within the Same Run:** Ensures there are no repeated questions within a single questionnaire run, counting and reporting the number of unique questions.
- **Cross-Questionnaire Unique Question Check:** Verifies the uniqueness of questions across different questionnaires for the same role and experience level.
- **Questionnaire Length Verification:** Confirms that each questionnaire consists of exactly 15 questions.
- **Questionnaire Generation Time Measurement:** Calculates and logs the time taken for the generation of each questionnaire.

The data obtained through the script was structured into an Excel spreadsheet. The initial columns (1-7) were automatically generated by the script, while three additional columns (8, 9, and 10) were manually filled in during the analysis phase.

Excel Spreadsheet Structure:

1. Profile: Represents the 6 technical areas (e.g., DevOps, Cloud Computing, Networking).
2. Seniority: Indicates the level of expertise required for each question (e.g., Junior, Mid-Level, Senior).
3. Number of the run: Specifies the iteration or batch number for each questionnaire generation process.
4. Question: Lists the generated questions for assessment.

5. Correct Answer: Provides the correct answer for each question.
6. Correct Answer is the Longest: A boolean value indicating whether the correct answer is the longest among the options.
7. Execution Time(s): Records the time taken to generate the questionnaire.
8. Correct Question: Used to verify if each question makes sense logically and grammatically.
9. Topic: Categorizes each questionnaire into specific topics or themes (e.g., Security, Database Management).
10. Seniority Level: A boolean value indicating whether the assigned seniority level was correct. If False, the correct seniority level was manually specified for each question.

After running the script and populating the entire Excel sheet, a final analysis was conducted based on metrics defined by our product owner (PO). Metrics to evaluate include:

- Seniority levels accuracy
- Percentage of correct questions/answers
- Percentage of appropriateness of questions to the seniority level
- Percentage of uniqueness of questions within the same test
- Percentage of uniqueness of questions across all tests (by profile and level)
- Percentage of uniqueness of tests (regardless of order)
- Diversity of topics per test
- Percentage of correct answer is not the longest by default
- Average response time from VertexAI (in seconds)

Here's the image presenting the results obtained from the metrics:

Note that the "Goal" column represents the expected results defined by our PO. These goals serve as benchmarks for evaluating the performance of the model against predefined standards. They provide a clear indication of the desired outcomes and help assess whether the model meets, exceeds, or falls short of the expected targets.

Here's a breakdown of the results:

Metric	Goal	Current	Meets
Seniority Levels	2.5	3	Yes
% Correct Questions/Answers	95	89.95	No
% questions pertaining to seniority	75	100	Yes
% unique questions in the same test	100	100	Yes
% unique questions in all tests (by role and level)	10	99.33	Yes
% single tests (regardless of order)	90	100	Yes
Different topics by tests	4	11	Yes
% Right answer is not the longest	65	62	No
Vertex AI Average Response Time (seconds)	10	22.065	No

Table 2: Feasibility Analysis.

- **Seniority Levels:** This metric compares the established goal (2.5) with the performance obtained (3) in accurately distinguishing between different levels of seniority. The value "3" corresponds to the successful differentiation of seniority levels across all areas. The result meets the established goal.

- **Percentage of correct questions/answers:** The goal was to achieve a 95% accuracy rate in correct questions and answers. However, the current performance falls short of this target, with a score of 89.95%. The percentage of correct questions/answers for each technical area varied. While some areas, such as Cybersecurity and React, achieved 100% correctness in some levels of seniority, others, such as Java, iOS, and Python showed the poorest results, with none of the three seniority levels reaching 95% accuracy in correct questions/answers within these domains. This indicates a need for improvement in question accuracy in this domains.

- **Percentage of appropriateness of questions to the seniority level:** The goal was to ensure that questions were appropriate for the designated seniority level, with a target set at 75%. The current performance meets this goal, achieving a score of 100%, indicating that questions are tailored appropriately to the seniority level.

- **Percentage of uniqueness of questions within the same test:** The goal was to ensure that questions were appropriate for the designated seniority level, with a target set at 75%. The current performance meets this goal, achieving a score of 100%, indicating that questions are tailored appropriately to the seniority level.

- **Percentage of uniqueness of questions across all tests (by profile and level):** The goal was to ensure diversity and uniqueness of questions across all tests, with a target set at 10%.

This target indicates that within each test generated, no more than 10% of the questions should be repeated from previous tests. However, there was a small error in the initial target setting, as it should have been set at 90% instead of 10%. This means that at least 90% of the questions should be unique across all tests. The current performance significantly exceeds this corrected goal, with a score of 99.33%. This score indicates that only 0.67% of the questions were repeated across the tests. In other words, the diversity and variety of questions across different roles and levels exceeded expectations, with a high degree of uniqueness in the generated quizzes.

- **Percentage of uniqueness of tests (regardless of order):** The aim was to ensure the presence of unique tests, with different questions across various tests within the same domain, regardless of the order of questions. This target was set at 90%. The current performance successfully meets this objective, scoring 100%.

- **Diversity of topics per test:** The goal was to ensure that each test encompasses a variety of topics, with a target of covering at least four distinct subject areas for questionnaire. The current performance exceeds this goal, with a score of 11, indicating a broad coverage of topics in each assessment.

- **Percentage of correct answer is not the longest by default:** The goal was to ensure that the correct answer is not consistently the longest option, with a target set at 65%. However, the current performance slightly misses this target, achieving a score of 62%. While it falls short of the goal, it's worth noting that this criterion may not be of highest priority.

- **Average response time from VertexAI (in seconds):** The goal was to maintain a fast response time from Vertex AI, with a target set at 10 seconds. However, the current performance exceeds this target, with an average response time of 22.065 seconds, indicating a need for optimization to reduce response times. While it's twice the goal, it's important to note that waiting around 20 seconds for quiz generation isn't considered excessive.

Overall, the assessment indicates a satisfactory performance in generating quizzes for various technical areas and seniority levels. The encouraging outcomes suggest a solid foundation for advancing the application's development, providing valuable insights for refining and improving the quiz generation process.

Furthermore, to validate the effectiveness of the quiz generation model, a final assessment was conducted using a specific job description as a test case. The evaluation aimed to thoroughly assess the

model's performance and relevance in evaluating candidates' technical skills and knowledge with minimal recruiter involvement. Three sets of identical tests, each consisting of 15 questions, were administered for evaluation purposes:

1. Set without Examples: No specific examples provided.
2. Set with Generic Programming Example: Includes a generic programming question, "What is the worst-case time complexity of a quick sort algorithm?" with multiple-choice options.
3. Set with Regular Senior Java Examples: Contains questions tailored to Java 11, Docker, SpringBoot, React, AWS, and Cloud technologies.

Each set was replicated three times to ensure comprehensive assessment and comparison across different scenarios. The context for this evaluation was a senior candidate profile with expertise in Java 11, Docker, SpringBoot, React, and some knowledge of Amazon Web Services (AWS) and cloud technologies.

An analysis was conducted to evaluate the distribution of questions related to these technologies across each set, as well as the inclusion of broader cross-technology or specific technology topics. The percentage of questions pertaining to each technology in each set was calculated, considering each test comprised 15 questions. This analysis provided insights into how well the generated quizzes covered the technical areas specified in the job description.

The table below presents the results obtained from this evaluation:

	Without example questions			With 1 generic example question			With the Java Senior questions		
Technologies / No. of questions	**Min**	**Max**	**%**	**Min**	**Max**	**%**	**Min**	**Max**	**%**
Java 11	3	3	20	0	4	14	3	7	31.11111111
Docker	2	3	17.77778	1	4	16	3	4	22.22222222
SpringBoot	3	4	22.22222	3	8	30	1	3	15.55555556
React	3	4	22.22222	2	4	18	2	3	17.77777778
AWS	2	3	15.55556	2	4	18	0	3	11.11111111
Cloud	0	0	0	0	0	0	0	0	0
Cross-tecnology/Generic	0	1	2.222222	0	1	2	0	1	2.222222222
Other Specific Technology	0	0	0	0	1	2	0	0	0

Table 3: Distribution of Questions Across Technology Areas.

The analysis of the results reveals several insights into the effectiveness of the quiz generation model in assessing candidates' technical skills and knowledge based on specific job descriptions. Overall, we observed variations in the distribution of questions across different sets, highlighting the adaptability of the model to generate diverse tests tailored to specific contexts.

In particular, the sets with generic example questions and senior Java questions demonstrated a wider range and higher frequency of technology-related questions compared to the set without example questions. This indicates that providing examples, whether generic or specific, can influence the diversity and depth of the generated quizzes, potentially enhancing their effectiveness in evaluating candidates' proficiency in relevant technologies.

Furthermore, the presence of questions related to cross-technology/generic topics suggests that the model can identify and include interdisciplinary concepts, reflecting the interconnected nature of modern technology stacks. However, the relatively low frequency of such questions indicates room for improvement in capturing and incorporating cross-technology knowledge into the generated quizzes.

Despite the overall promising results, some areas, such as and Cloud technologies, exhibited limited representation in the generated quizzes. This highlights the importance of fine-tuning the model to ensure comprehensive coverage of all specified technologies in job descriptions, thereby providing a more holistic assessment of candidates' technical competencies.

In conclusion, while the quiz generation model shows potential in efficiently generating tailored assessments, further refinement is needed to optimize the balance and coverage of technology-specific questions, particularly in niche domains.

4.5 Architecture Solution

The architecture design comprises several components seamlessly integrated to achieve the project's objectives. This design follows the Model-View-Controller (MVC) pattern, which ensures a clear separation of concerns, facilitating maintenance and scalability of the application.

The frontend is developed using SvelteKit, a modern web framework that utilizes HTML and CSS for creating user interfaces. SvelteKit offers a streamlined development experience, enabling efficient rendering of dynamic web content and ensuring optimal performance for end-users. The project employs Tailwind CSS, allowing for rapid styling and consistent design by composing styles directly in the markup. In addition, daisyUI is integrated to enhance the styling process. DaisyUI is a plugin for Tailwind CSS that adds a set of customizable, pre-designed components and themes, significantly speeding up the development process. It provides a rich set of ready-to-use UI components and built-in themes, making it easier to maintain a consistent design language across the application.

For the backend infrastructure, Python is employed as the programming language. Flask, a micro web framework for Python, is utilized to build the backend services. Flask is known for its simplicity and

flexibility, making it an excellent choice for this application. It provides the essential tools and libraries needed to build web applications, including routing, request handling, and integration with databases. Flask's lightweight nature allows developers to have more control over the components they use, leading to efficient and scalable backend services. The backend contains the application logic, processing user inputs from the frontend, and interacting with the database and external services.

The Firebase Realtime Database is utilized as the data storage and synchronization solution. Firebase offers a scalable, cloud-hosted database service that allows real-time data synchronization across clients and devices. By leveraging Firebase Realtime Database, the project ensures efficient data management, synchronization, and accessibility, enabling seamless collaboration and interaction between users. This database automatically updates data across all clients in real-time, providing a consistent and up-to-date user experience. To leverage ML capabilities, the project integrates with the VertexAI service API. VertexAI provides a comprehensive suite of tools and services for building, deploying, and managing ML models.

The image below illustrates the architecture following the MVC pattern:

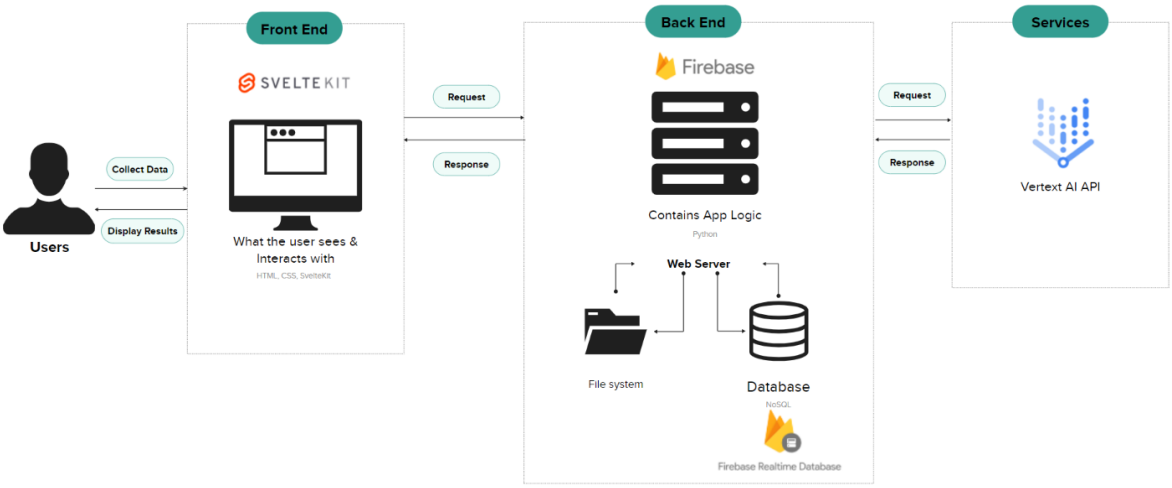


Figure 9: System Architecture.

- Model: Firebase Realtime Database and VertexAI API handle data storage and external services.
- View: The SvelteKit frontend provides the user interface, presenting data and capturing user input.
- Controller: Backend application logic, implemented in Python, processes inputs from the view and interacts with the model.

Overall, the architecture design encompasses a frontend developed with SvelteKit, integrated with VertexAI API for ML capabilities, backed by Python for backend logic, and utilizing Firebase Realtime Database

for data storage and synchronization. This MVC-based architecture enables the creation of a robust, scalable, and efficient application that fulfills the project's requirements and objectives.

4.6 Domain Model

This domain model outlines the entities and their attributes within the recruitment system. The model includes various entities with defined roles and relationships:

- **User:** Serves as the base entity with common attributes like userId, email, first name, last name, password, and role (recruiter or candidate). Both recruiter and candidate roles are derived from this class, inheriting all its attributes.
- **Recruiter:** A specific type of User tasked with managing job openings, known as demands. A recruiter can handle numerous demands but maintains a unique association with each. They maintain a list of the demands they have created.
- **Candidate:** Also a specialized type of User, a candidate can be associated with multiple demands and can take multiple quizzes. Attributes include quiz list, job role, and candidate status (options include New, Ready, Pending, Failed, Sent and Done), illustrating the candidate's progression through different job applications and assessments.
- **Administrator:** A special case of User, the Administrator role is designed to supervise and oversee the entire system. Administrators can directly manage demands, which includes editing existing ones and deleting demands that are no longer active.
- **Demand:** Each demand is uniquely managed by a single recruiter and can include multiple Candidates. It contains attributes like projectName, clientName, jobDescription, jobRole, seniority, and deadline, demand status (indicating whether it is open or closed), defining the specifics of a job opening. For example, a recruiter might create separate demands for "Java Senior Developer" and "Java Junior Developer" positions.
- **Quiz:** Represents the assessments candidates must take, associated with specific demands. Each quiz is uniquely designed for an individual candidate, although a candidate may participate in several quizzes. It includes attributes like questions list, candidate answers, score, timeLimit, number of questions and quiz status (true or false).

The status of each entity will be explained later. For now, this is just to give a general overview of the basic functionality. Note that there is no direct relationship between recruiters and quizzes in the diagram. Quizzes are created and managed by an external system (Gemini model), based on the parameters specified by recruiters through the demands. This implicit relationship is crucial for understanding that while recruiters influence quiz content and structure, they do not directly create or manage quizzes.

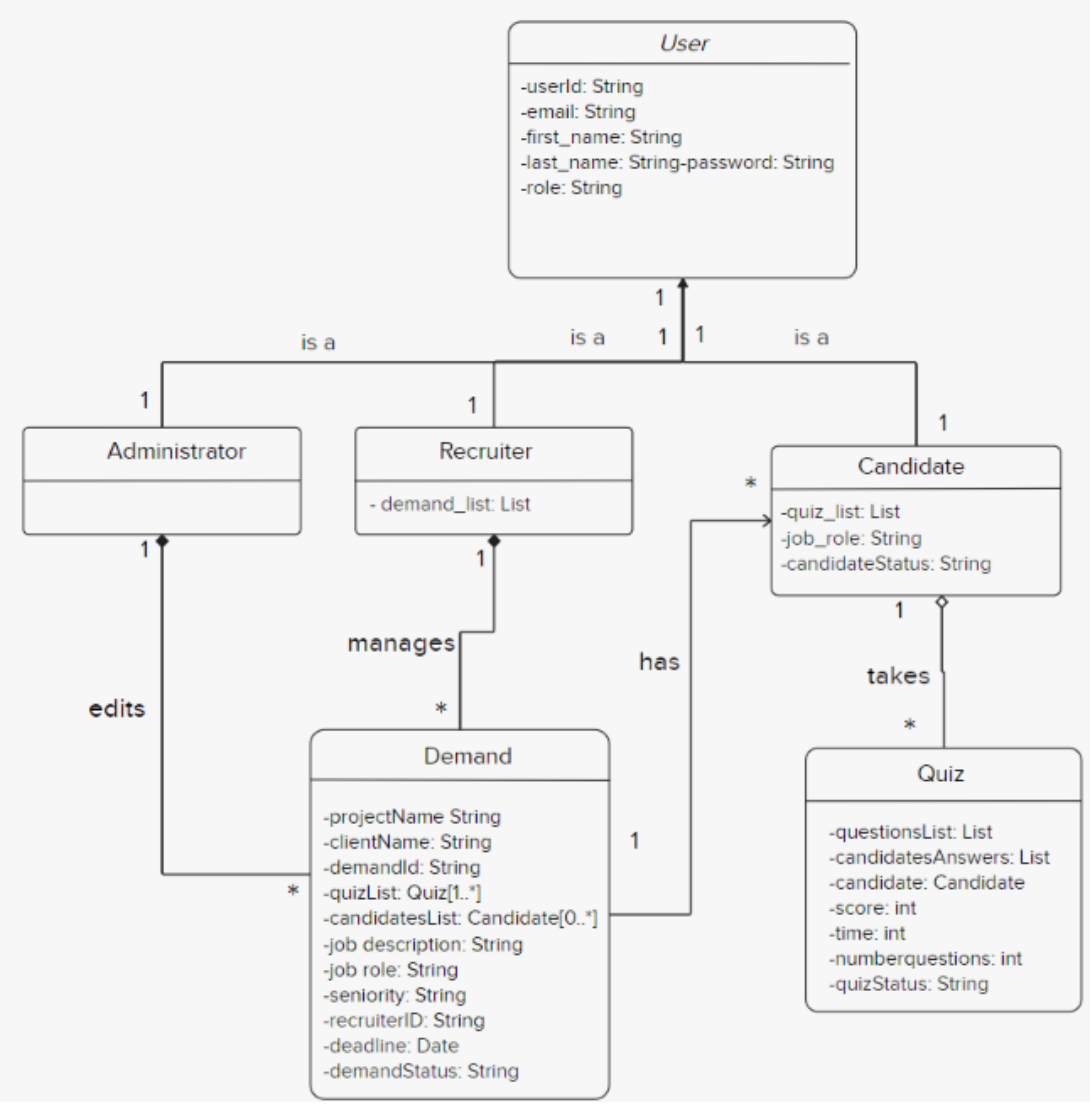


Figure 10: Domain Model.

4.7 Workflow

In this section, we explore the recruitment workflow expected on the application.

The process begins when a recruiter defines the job specifications and sets up a demand within the system. Recruiters can add candidates by inputting their personal data into the system.

After setting up the project and adding candidates, the recruiter can generate a questionnaire for the candidates. These questionnaires are automatically generated by the model, specifically the Gemini Pro 1.5, based on the job description provided.

The recruiter has the option to download the questionnaire as a PDF for review. This PDF includes all demand information, questions, answer options, correct answers, and candidate responses if the candidate has already completed the quiz.

Candidates access the questionnaire via a registration link provided by the recruiter and proceed to answer the questions as specified. Upon submission of the questionnaire, scores are automatically calculated and displayed as percentages. The recruiter can decide whether candidates can view their scores.

Recruiters monitor demand results through a statistics dashboard, which provides insights into each candidate's performance relative to the specific demand.

This streamlined system automates many aspects of the recruitment process, enhancing efficiency and accelerating candidate evaluation. Recruiters can then advance suitable candidates to subsequent hiring stages based on their performance.

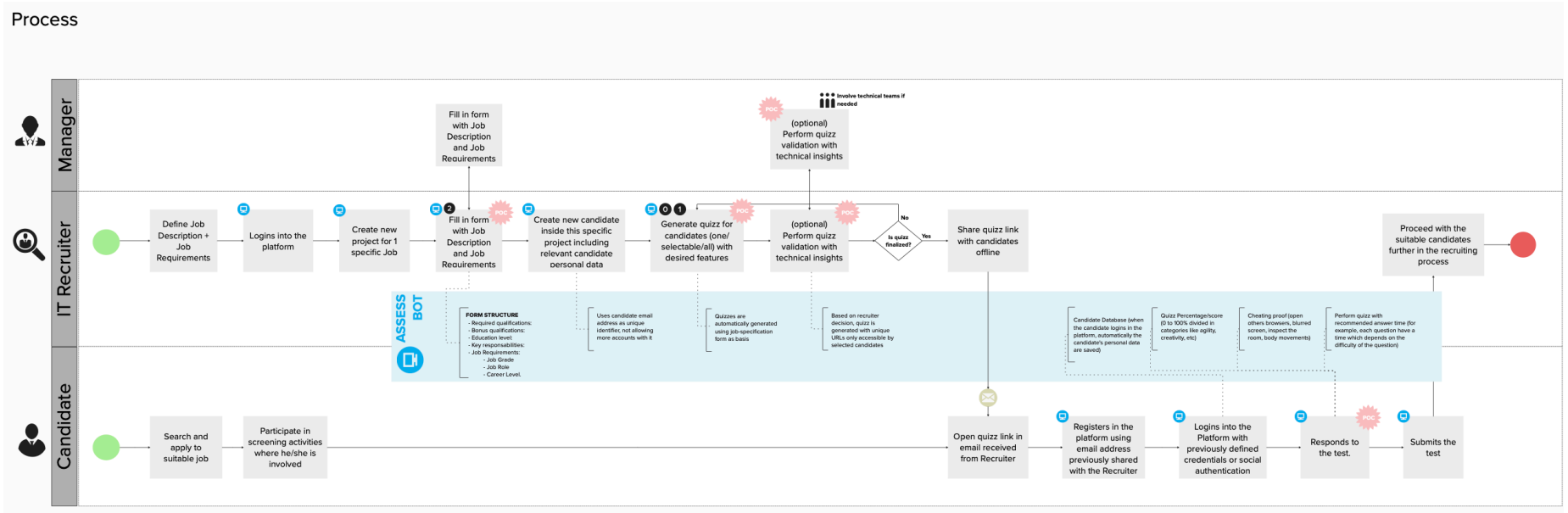


Figure 11: Workflow.

4.8 Database Structure

In this section, we will delve into the database structure tailored for our recruitment platform.

The database utilized for this dissertation encompasses various fields pertinent to the study, each meticulously organized to streamline data management and retrieval. The key fields within this database include:

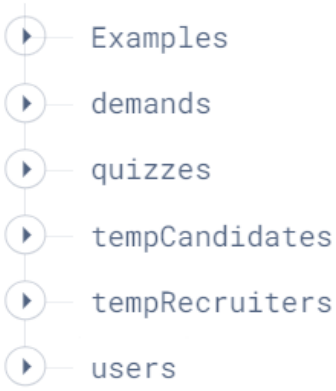


Figure 12: Database Structure.

Now, let's delve deeper into each of the fields.

4.8.1 Examples

The "Examples" field serves as a repository of illustrative instances, providing practical demonstrations and scenarios across different domains. Within this field, data is structured categorically, with subdivisions tailored to specific areas of interest. For instance, under the "cybersecurity" category, questions and corresponding answers are meticulously cataloged for varying proficiency levels – "Junior", "Mid", and "Senior". Each question is formatted with input prompts, multiple-choice options, and a designated correct answer. Similar structures are replicated for other fields such as Java, React, Python, iOS, and DevOps, ensuring comprehensive coverage and accessibility of knowledge across diverse subjects.

- Examples
  - Cybersecurity
    - Junior: "input: ... output:... Correct answer: ..."
    - Mid

        * Senior
    - Devops
    - Ios
    - Java
    - Python
    - React

4.8.2 Demands

The "Demands" section encapsulates the core requirements and details for each recruitment project initiated within the system. Each demand is meticulously structured to define the specific job role, client information, project details, and recruitment parameters. Below is an outline of the typical structure found within the "Demands" dataset:

- Demands
    - demandID: Unique identifier for each demand.
        * candidates: Collection of candidates associated with the demand.
            - candidateID: Unique identifier for each candidate.
            - completed_at (%Y-%m-%d %H:%M:%S): Date and time when the quiz was completed.
            - email: Candidate's email address.
            - first_name: Candidate's first name.
            - last_name: Candidate's last name.
            - quiz_id: Identifier of the quiz completed by the candidate.
            - score: Candidate's score on the quiz.
            - status: Current status of the candidate in relation to the demand.
        - clientName: Name of the client associated with the demand.
        - deadline: Deadline by which the demand needs to be fulfilled.
        - demandStatus: Current status of the demand (active, pending, closed, etc.).
        - jobDescription: Detailed description of the job requirements.
        - jobRole: Specific role or position being recruited for.

- projectName: Name of the project associated with the demand.
- recruiterId: Identifier of the recruiter managing the demand.
- seniority: Desired level of seniority for candidates (junior, mid-level, senior, etc.).

– demandID

4.8.3 Quizzes

The "Quizzes" field houses a collection of interactive assessments tailored to evaluate proficiency and understanding across various domains. Below is an example of how a quiz is structured within the database.

- Quizzes

  – quizID: Unique identifier for each quiz.

    - Questions
      - 0: first question
      - Options
      - 0: option 1
      - 1: option 2
      - 2: option 3
      - CorrectAnswer: Indicates the text of the correct option for the question.
      - Question: The text of the question being asked.
      - 1: second question
      - 2: third question
    - answers
      - 0: Selected answer for question 1.
      - 1: Selected answer for question 2.
      - 2: Selected answer for question 3.
    - access_to_score (boolean): Determines if candidates can view their scores post-quiz.
    - completed_at (%Y-%m-%d %H:%M:%S) Records the date and time when the quiz was completed.

        - expected_time (minutes): The anticipated duration to complete the quiz.
        - job_description: Description of the job role associated with the quiz.
        - job_role: Specific role being assessed by the quiz.
        - option_of_backwards (boolean): Indicates if candidates can navigate backward during the quiz.
        - score (percentage): Candidate's score achieved in the quiz.
        - seniority: Level of seniority for which the quiz is intended (e.g., junior, mid-level, senior).
        - status (boolean): Indicates if the quiz is active or inactive. A status of false denotes that the candidate has completed the quiz.
    - quizID

The "answers" field records the options selected by candidates during the quiz. Each candidate must answer all questions to submit the quiz. However, if a candidate does not answer a question, it is typically because they ran out of time, resulting in the quiz automatically submitting with any unanswered questions recorded as "Did not answer."

4.8.4 TempCandidates

TempCandidates serve as placeholders for individuals who have yet to formalize their presence within the application. When a recruiter associates a candidate with a demand, a unique identifier is generated for that candidate and added to the TempCandidates repository. Here, pertinent candidate information is stored temporarily, awaiting the candidate's official registration. Upon registration, all information residing within TempCandidates is seamlessly transferred to the Users repository, effectively transforming the candidate from a temporary entity to an official user of the application. This mechanism ensures a smooth transition from candidate status to full-fledged user, consolidating all relevant data under a unified user profile.

- tempCandidates
    - candidateID: Unique identifier for each candidate.
        - demandList: List of demand IDs associated with the candidate.
            - 0: "demandID"

        - 1: "demandID"

    - email: Candidate's email address.
    - first_name: Candidate's first name.
    - last_name: Candidate's last name.

  - candidateID

4.8.5 TempRecuiters

Similarly, TempRecruiters serve as a registry for recruiters who have yet to complete the registration process within the application. When a recruiter initiates registration, their identifier is temporarily stored in TempRecruiters. This repository acts as a safeguard against unauthorized entities attempting to masquerade as recruiters within the application. By maintaining a record of recruiter identifiers, the application can preemptively identify and authenticate legitimate recruiters, thereby fortifying the integrity of its user base. Once registration is completed, the recruiter's information is seamlessly transitioned to the Users repository, signifying their official status within the application.

- tempRecruiters
  - recruiterID: Unique identifier for each recruiter.
    - demandList: List of demand IDs associated with the recruiter.
      - 0: "demandID"
      - 1: "demandID"
    - email: Recruiter's email address.
    - first_name: Recruiter's first name.
    - last_name: Recruiter's last name.
  - recruiterID

4.8.6 Users

The "Users" field encapsulates user profiles and authentication credentials essential for database access and interaction. Each user is assigned a unique identifier and associated metadata, facilitating personalized access permissions and tailored user experiences within the database environment.

In essence, TempCandidates and TempRecruiters serve as crucial intermediaries in the journey from prospective user to official member within the application's ecosystem. Through their seamless integration and data transfer mechanisms, these entities ensure the integrity, security, and efficiency of the user registration process, safeguarding the application against fraudulent activities and fostering trust among its user community.

The administrative role is managed differently compared to TempCandidates and TempRecruiters within the system. Unlike temporary candidates and recruiters, the administrator is directly added to the database with a unique identifier. Only essential information is stored for the administrator, including their email, first name, last name, and role (admin). Unlike TempCandidates and TempRecruiters, the administrator user does not maintain a demandList field. This distinction reflects their role, which focuses on overseeing system administration and management rather than participating in specific recruitment processes or projects within the application.

- Users
  - userID: Unique identifier for each registered user.
    - demandList: List of demand IDs associated with the user.
      - 0: "demandID"
      - 1: "demandID"
    - email: User's email address.
    - first_name: User's first name.
    - last_name: User's last name.
    - role: User's role within the application.
  - userID

4.9 Platform Design

This section is divided into three parts: recruiters, candidates, and admin. We'll explore each view in detail, presenting images obtained from the platform that showcase the user interface and functionality designed specifically for each role. Each image illustrates specific features and interactions aimed at streamlining recruitment processes and improving the job application experience. The platform's style and colors are designed to align with Accenture's branding, prominently featuring the color purple to match the company's logo.

The homepage serves as the entry point for users with a "Welcome to AssessBot" message, offering options to log in or register in the upper right corner. Depending on whether users choose to log in or sign up, they are directed to the corresponding pages as illustrated in the figures provided.

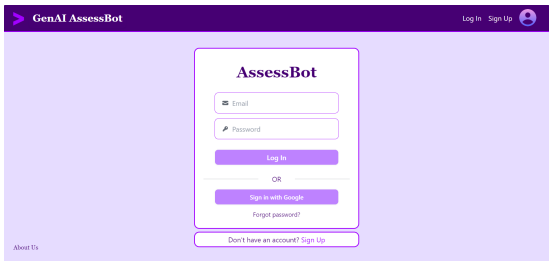


Figure 13: Login page.

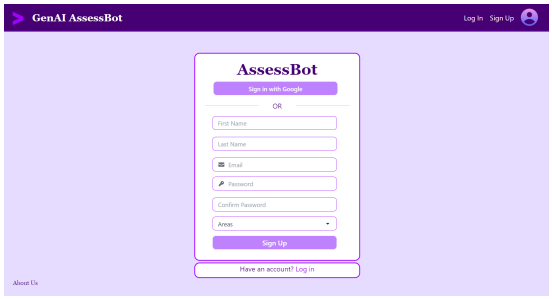


Figure 14: Sign up page.

Users can log in using their email and password or opt for Google login. For those opting for Google login, a Google authentication pop-up appears, facilitating a quick and secure login process using their Google credentials.

To register, users must provide their first name, last name, email, and password. Optionally, users can also specify their area of expertise during the registration process.

4.9.1 Recuiters

First, we will explore the recruiter's viewpoint within the application, detailing their interface and the functionalities available to them for managing the recruitment process.

Upon logging in, recruiters are greeted with a "Welcome, Recruiter" message at the center of the screen. Additionally, in the upper right corner, there's a personalized greeting that says "Welcome" followed by the recruiter's first name. To the left of the screen, recruiters find two buttons: "My Demands" and "My Dashboard". Clicking on "My Demands" displays a list of demands created by the recruiter.

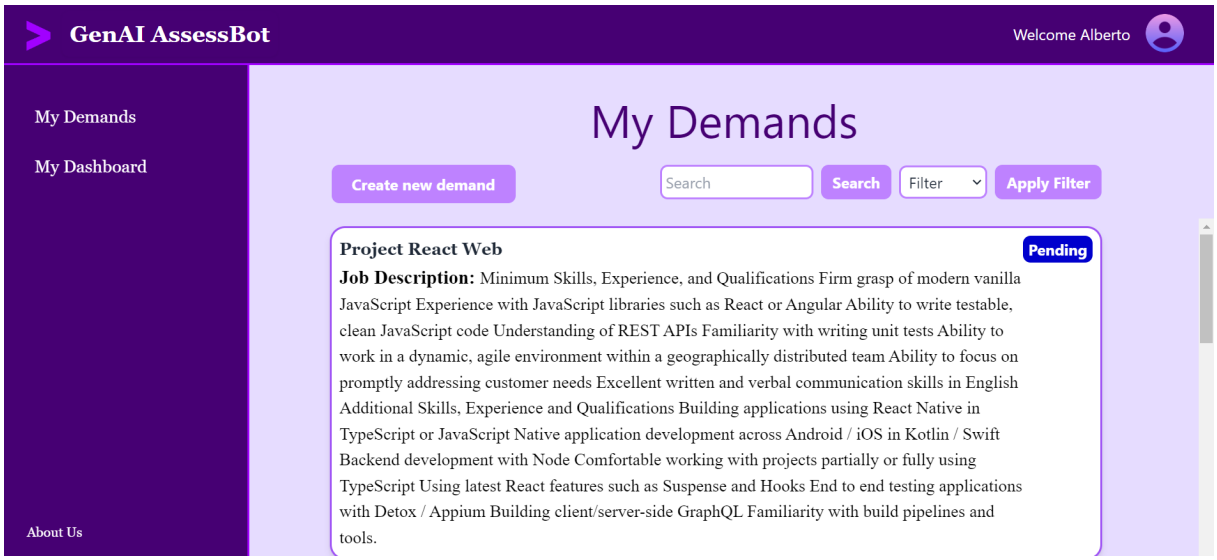


Figure 15: Demands list page.

Recruiters can create a new demand by clicking on the "Create new demand" button and filling out fields such as project name, client name, fulfillment deadline, job role, job description, and seniority. After creating a demand, a pop-up appears indicating that the demand was successfully created, and the recruiter is redirected to the "My Demands" page. The demand ID is asynchronously added to the database. It is added to the "demands" table with all the demand-related information and also to the "demandList" field under the recruiter's ID in the "users" table.

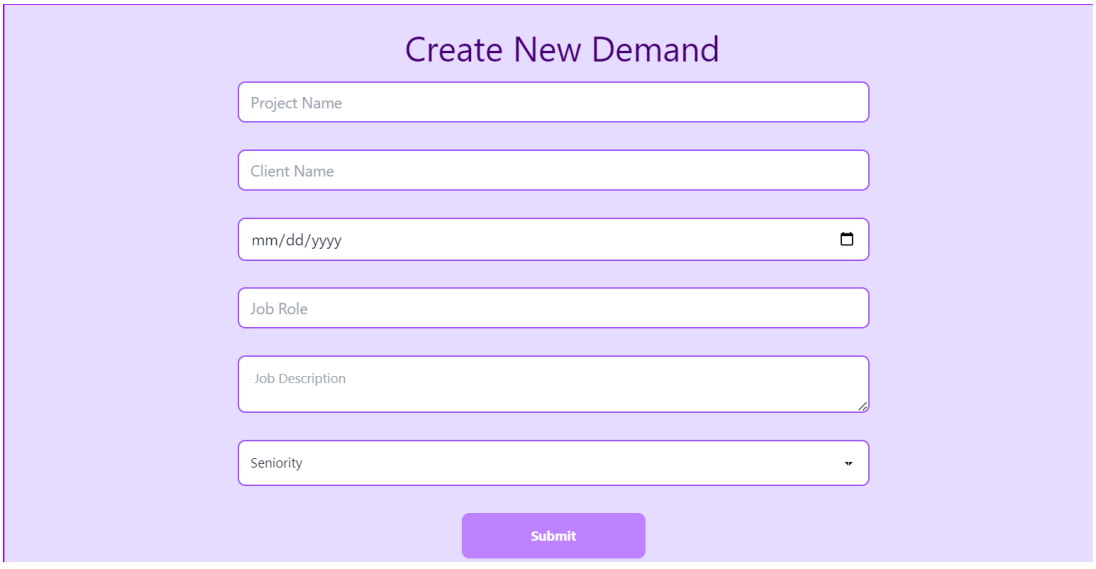


Figure 16: Create Demand page.

It is also possible to filter demands to display only those in a desired state. The states of demands,

on the recruiter side, can be categorized as follows:

- Open: The demand has been created and is waiting to be filled by candidates.
- Pending: Candidates have been added to the demand.
- Closed: The demand has been completed, either by the recruiter closing it manually or by reaching its deadline.

Clicking on a specific demand reveals detailed information including title (project name), client name, fulfillment date, job role, job description, seniority, list of associated candidates, and options to generate or import new candidates. Recruiters can filter by states and also have the option to select all candidates. Additionally, recruiters have the capability to close a demand and transfer ownership of a demand to another recruiter.

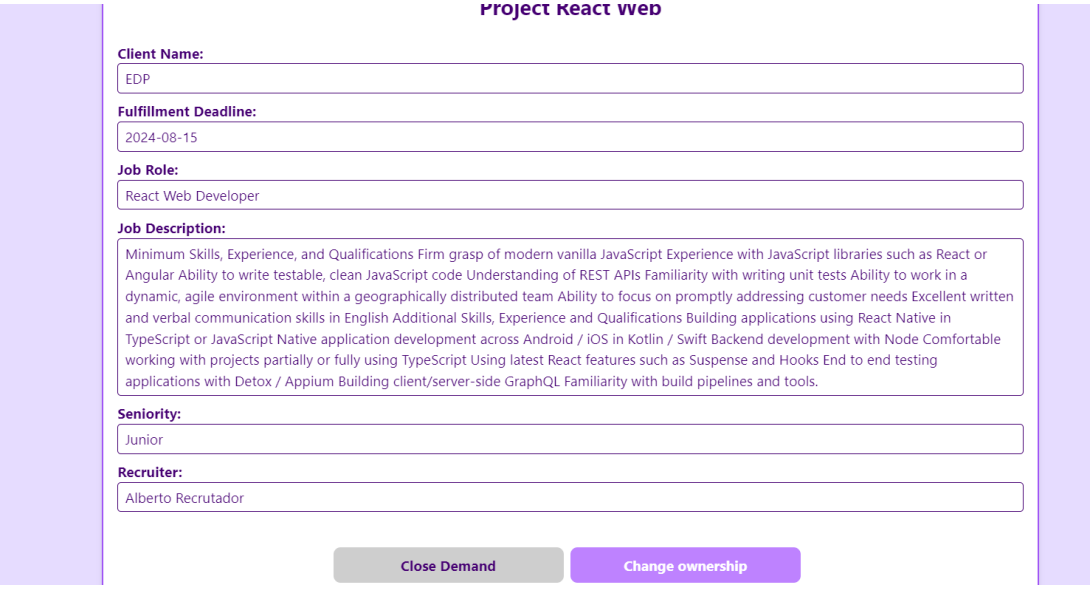


Figure 17: Demand page.

For candidates, the states change as follows:

- NEW: Candidate has been added to the demand.
- PENDING: Quiz request sent to AI.
- FAILED: Quiz generation failed after three attempts. This occurs when the model unsuccessfully tries to generate a valid quiz, failing steps of the validation script. In such cases, the recruiter has the option to attempt generating a new quiz for the candidate. Detailed information about the validation script will be discussed in the following section.

- READY: Quiz generated.
- SENT: Email sent to the candidate. This email notifies candidates that they have a quiz associated and provides a link to the registration page of the application, not directly to the quiz.
- DONE: Quiz responded by candidate.

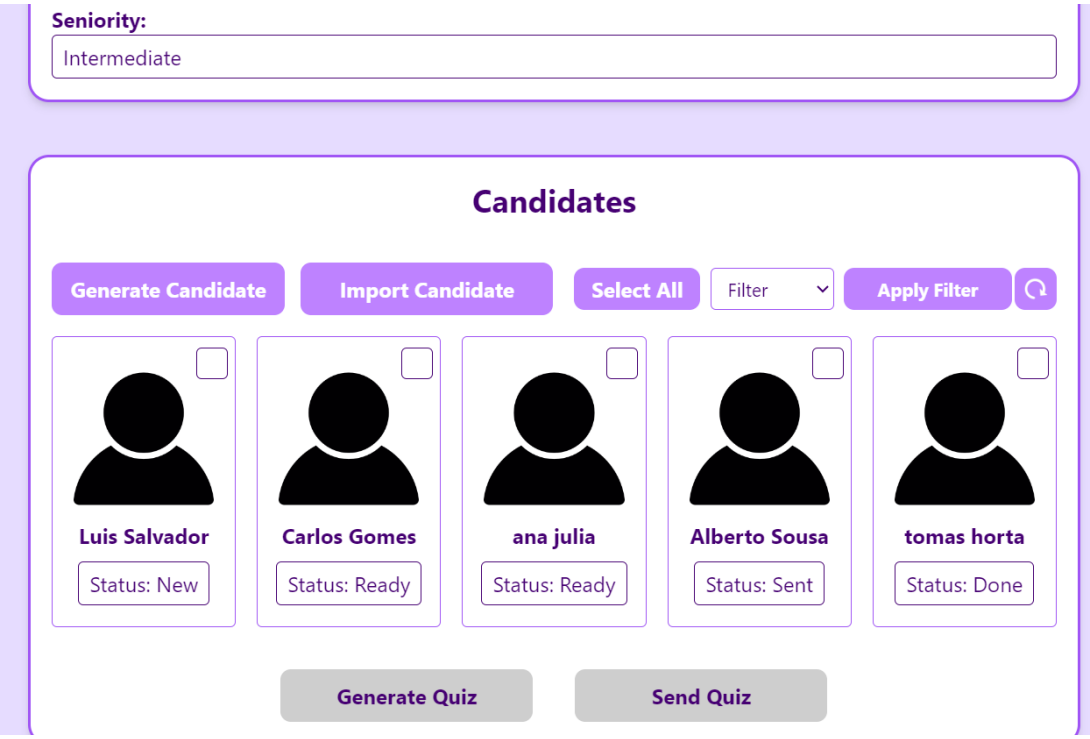


Figure 18: List of candidates associated with the demand.

Recruiters can import candidates by clicking on the "Import Candidate" button, which opens a file explorer window where the recruiter can upload a file containing the candidates' information, including first name, last name, and email. Additionally, there is a "Generate Candidate" button for manually adding a candidate. Clicking this button opens a pop-up where the recruiter must fill in the first name, last name, and email fields, and then click "Generate Candidate" to add a single candidate manually. Once added, candidates appear with the status "NEW." It is also possible to select all candidates by clicking "Select All" or filter candidates by their status using the "Filter" option.

Recruiters can generate quizzes for selected candidates by clicking on the "Generate Quiz" button, which opens a pop-up window where they can specify parameters such as the number of questions, expected time, access to scoring, and option for backwards questions through the interface.

After generating a quiz, if it does not fail, the status appears as "READY." The "Generate Quiz" button is available in all states except "SENT" and "DONE." In the "READY" state, recruiters can regenerate the

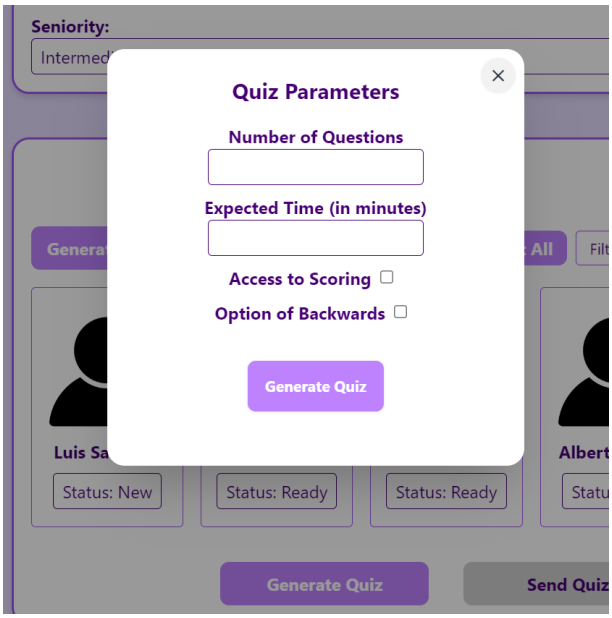


Figure 19: Quiz parameters.

quiz, which will replace the previous quiz with a new one. For candidates in the "NEW" status, it indicates that a quiz has not yet been generated for them, hence clicking the "Send Quiz" button is not possible. However, candidates in the "READY" status allow recruiters to send the quiz via email by clicking the "Send Quiz" button. This action triggers a backend function via a POST request, which utilizes Simple Mail Transfer Protocol (SMTP) for email dispatch.

It's important to note that our system does not employ an SMTP server directly due to cost considerations. Instead, this process was simulated in our application for demonstration purposes. When the "Send Quiz" button is activated, the system simulates the email dispatch by updating each candidate's status to "Sent" in the database and notifying the recruiter via a pop-up confirmation that the quiz has been successfully sent.

Recruiters can view the profile of a specific candidate by clicking on the candidate icon, accessing information such as full name, email, job role, status, quiz progress status, and quiz scores. In this page, the recruiter also has the option to generate a quiz, which functions similarly, displaying the quiz parameters in a pop-up for the recruiter to fill out. In this case, the selected candidate is only the candidate on the current page, meaning a quiz will be generated solely for this specific candidate.

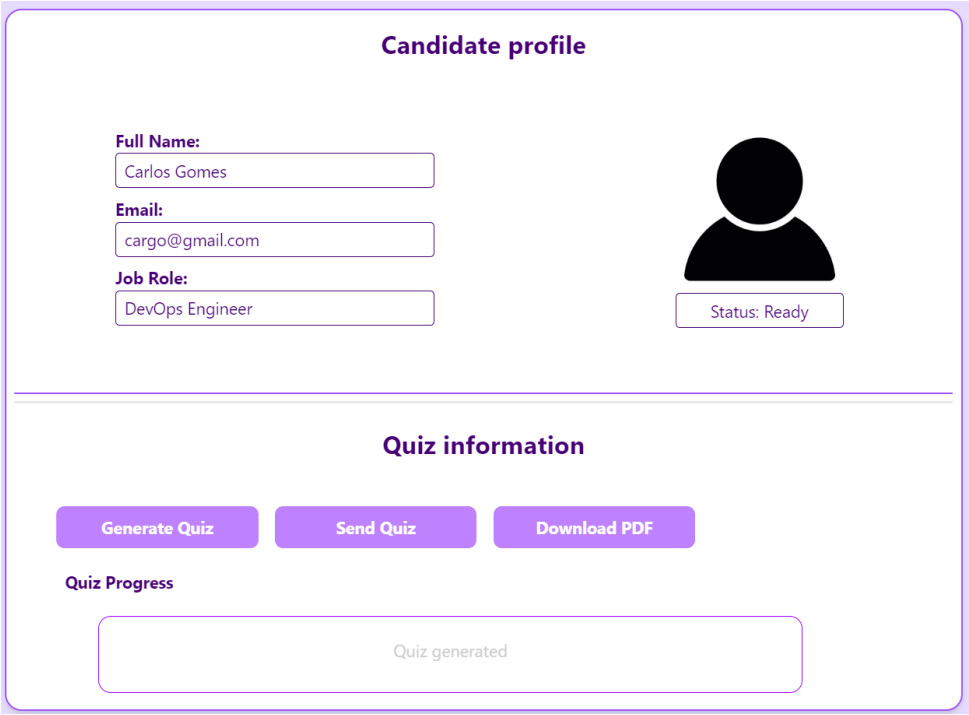


Figure 20: Carlos's profile.

Additionally, recruiters have the option to download a PDF of the quiz associated with the candidate, where it displays the correct answers with checkmarks and any incorrect selections with "X" marks. If the candidate's selected answer is correct, only the checkmark is displayed. This feature is useful for technical validation and review, providing a detailed overview of the quiz content, including questions, answer options, and the correct answers. The PDF download option ensures accessibility to quiz details at any stage, facilitating thorough assessment and evaluation.

The "My Dashboard" section allows recruiters to view detailed statistics for each demand, aiding in evaluating recruitment performance. Upon accessing the dashboard, recruiters can see various demand projects listed, each represented by a button.

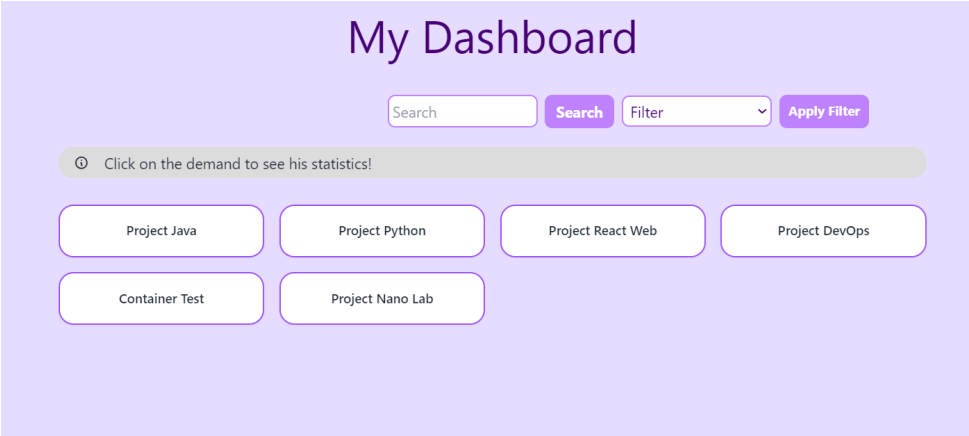


Figure 21: Statistics Dashboard.

Clicking on a demand displays several metrics, including:

- **Percentage of candidates who finished the quiz:** This metric shows the percentage of candidates who have completed the quiz. For example, 15.385% indicates that a small portion of candidates completed the quiz.
- **Top 3 Candidates:** This section lists the top-performing candidates with their respective scores. The candidate with the highest score appears first, followed by the next highest, and so on. The score is displayed in percentage next to the candidate's name. It helps recruiters identify the best candidates quickly.
- **Number of candidates:** Displays the total number of candidates associated with the demand.
- **Average score:** Shows the average score of the candidates that finished the quiz, providing a quick insight into overall candidate performance. For instance, if a job position has 10 candidates and only two of them have finished the quiz, thereby having a score associated with them, then the average would be calculated by dividing the sum of the scores of these two candidates by the total number of candidates who completed the quiz, in this case, two.
- **Median quiz score:** A histogram displays the distribution of quiz scores, allowing recruiters to see how candidates performed across different score ranges. This visualization highlights the most common score ranges and any outliers.
- **Number of candidates per status:** A bar chart indicates the number of candidates in each status (Done, Failed, New, Pending, Ready, Sent). This helps recruiters understand the current state of the recruitment process and identify any bottlenecks or areas requiring attention.

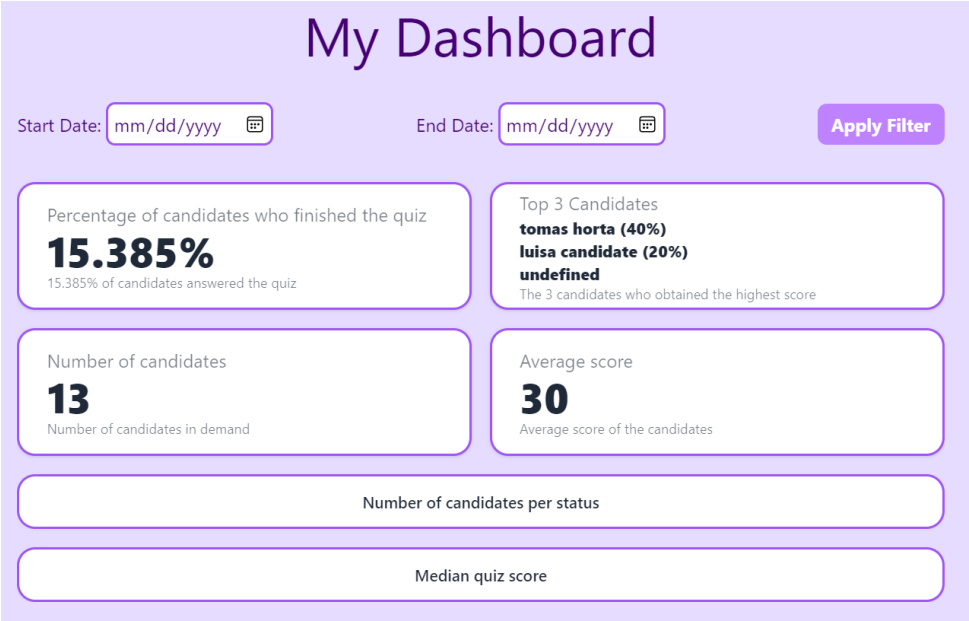


Figure 22: Dashboard for one specif demand.

There is also a calendar selector integrated in the dashboard of a specific demand for recruiters to select the start and end dates (year, month, and day). This feature allows recruiters to filter and view statistics within a specified time range. If no dates are selected, the dashboard displays statistics covering all candidates. It is important to note that only the "Percentage of Candidates who finished the quiz" and the "Top 3 Candidates" metrics are influenced by the selected date range. This is because the 'completed_at' field, which records the date and time when a candidate completes the quiz, is only associated with the 'Done' status of a candidate's response status.

4.9.2 Candidates

Now, we will explore the candidate's view within the application, detailing their interaction with the platform and the features available to them during the recruitment process.

When a candidate logs in, they are greeted with a "Welcome Candidate" message in the center of the page, and "Welcome [First Name]" appears in the upper right corner, similar to the recruiter interface. Unlike recruiters who have access to both "My Demands" and "My Dashboard" on the left side, candidates only have access to "My Demands." This section displays the demands to which they are associated.

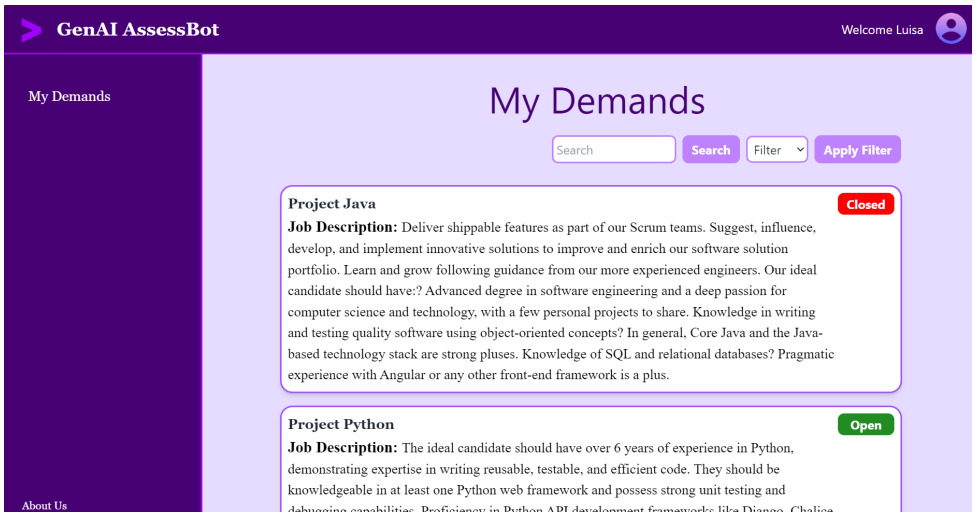


Figure 23: My demands page for candidates.

On the "My Demands" page, candidates can also filter demands by their status. In this case, there are two states: "open," indicating that the candidate is associated with the demand, and "closed," indicating that the quiz has been completed. Unlike the recruiter view, which includes three states (open, pending, and closed), the candidate view simplifies this to just two states.

Upon clicking on a specific demand, candidates can view information such as the title (project name), job role, job description, and seniority. If the demand is open, it means the candidate is associated with that demand, but it doesn't necessarily mean a quiz is available. If no quiz is associated, the message "You do not have access to this quiz yet!". If a quiz is available and pending, the "Respond to Quiz" button is active, and the status is displayed as "not completed." When the candidate clicks on "Respond to Quiz," a pop-up appears with instructions for the quiz, including the parameters set by the recruiter.

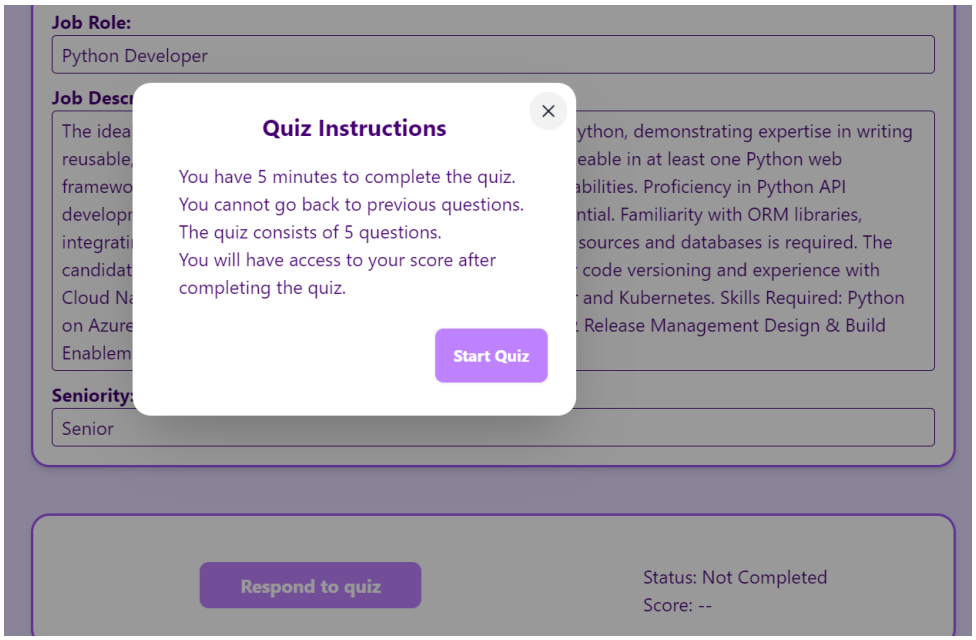


Figure 24: Quiz instructions.

If the candidate accepts, they click on "Start Quiz," and the quiz begins. Each question is displayed one at a time. In the lower right corner, the progress of questions (e.g., 1/5) is shown, and the remaining time to complete the quiz is displayed at the top. Next and previous buttons allow the candidate to navigate through the questions. The previous button may be disabled if the recruiter has decided that candidates cannot go back to previous questions. On the last question, the "Next" button is replaced by "Submit." Candidates must select an answer before being able to click "Next" and proceed to the next question. A pop-up message appears if no answer is selected, stating "Please select an answer before moving to the next question." If the time runs out, the quiz is automatically submitted, and a message appears stating "Time has run out!".

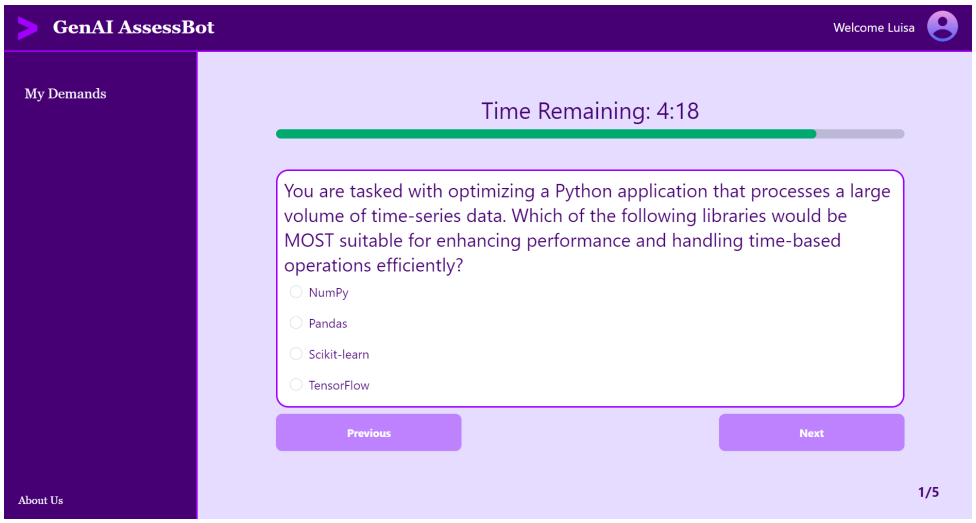


Figure 25: Quiz.

When the demand status changes to "closed," indicating that the quiz has been completed, the "Respond to Quiz" button becomes unclickable, appearing grayed out. The status is displayed as "completed," and if the recruiter has allowed candidates to view their scores, the score obtained by the candidate is displayed.



Figure 26: My Demands page for candidates with a closed demand.

4.9.3 Administrator

The administrator role in an application serves as a key overseeing position, responsible for managing and controlling the entire recruitment process from a macro perspective.

As we can see in the figure below, the administrator has a comprehensive toolkit to manage all demands effectively.

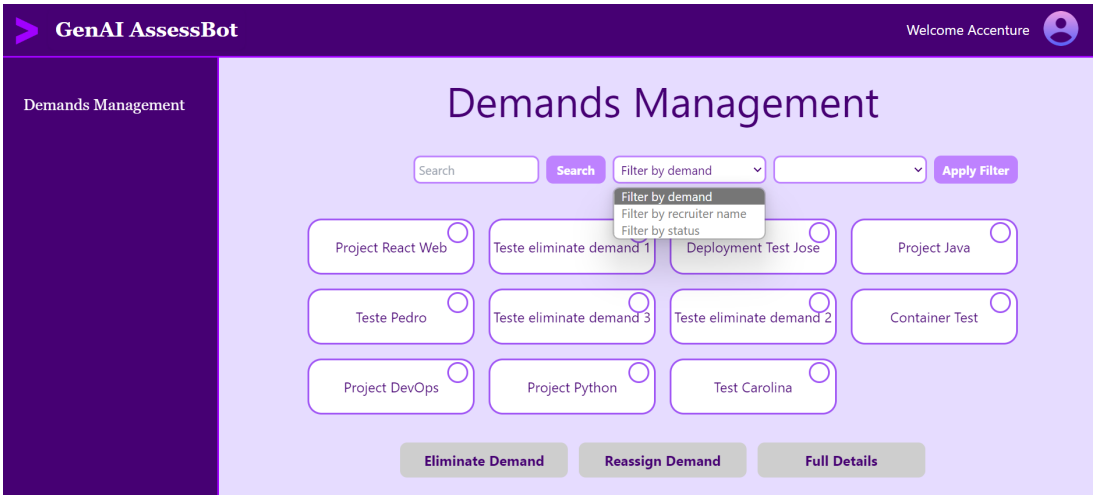


Figure 27: Demands management - Admin page.

At the top, there are search and filter functionalities that allow the administrator to quickly locate specific demands. They can filter demands by specific demand names, by the recruiter's name managing the demand, or by the current status of the demand (open, pending, closed). This makes it easier to oversee and manage large volumes of demands without having to manually search through each one.

Demand Management Actions:

- **Eliminate Demand:** The administrator can remove demands that are no longer needed or have been completed. This helps in keeping the database clean and up-to-date.
- **Reassign Demand:** If a recruiter is no longer available or if workload balancing is needed, the administrator can reassign demands to another recruiter. This is crucial for maintaining workflow continuity.
- **Full Details:** By selecting a demand and clicking "Full Details," the administrator can view comprehensive information about any demand. This deep dive helps in making informed decisions and adjustments. The page layout is similar to the ones shown in Figure 17 and Figure 18.

4.10 Script for validation of generated questionnaires

Since the quizzes are generated automatically by the Gemini model, it is essential to have an automated validation process to ensure certain aspects of the quizzes meet the required standards. The script is meticulously designed to automate critical validation processes, ensuring the adherence of generated quizzes to predefined standards and specifications. By implementing a series of validations, the script carefully examines various aspects of the generated quizzes to guarantee their integrity and quality. These validations serve to verify that the output, representing quizzes, is meticulously formatted and aligns with all required specifications for each quiz. This proactive approach to validation not only streamlines the quality assurance process but also enhances the reliability and consistency of the generated quiz content. Here's a brief explanation of the validations it conducts:

- **Number of Questions Validation:** The script checks if the number of questions in the generated quiz matches the specified number of questions parameter (See Figure 19). This validation ensures that each quiz contains the expected number of questions, preventing discrepancies in quiz lengths.
- **Uniqueness of Questions Validation:** It verifies whether each question in the quiz is unique by comparing it against the set of all previously encountered questions. This validation ensures that no duplicate questions are included in the quiz, maintaining diversity and avoiding redundancy.
- **Number of Answers per Question Validation:** For each question in the quiz, the script checks if the number of provided answers is exactly four. This validation ensures consistency and adherence to the quiz format, as quizzes typically present four answer choices per question.
- **Presence of Correct Answer Validation:** It confirms that each question has a correct answer among the provided choices. This validation ensures quiz accuracy and effectiveness by ensuring that candidates can accurately respond to the questions.
- **ASCII Character Validation:** The script includes an additional validation to ensure that the output generated by the model consists entirely of ASCII characters. This validation aims to address previous instances where non-ASCII characters were inadvertently included in the quiz data.

By implementing these validations, the script ensures that the generated quizzes meet predefined criteria for quality, consistency, and compatibility. This comprehensive approach enhances the reliability and usability of the quiz generation process, contributing to a seamless user experience for both recruiters and candidates.

4.11 Price estimation and Deployment

In this section, we will discuss the estimation of deployment costs. The deployment process involves estimating the cost to ensure budget alignment and selecting the appropriate deployment method for the application. We conducted three attempts to deploy the application: Cloud Run, App Engine, and Kubernetes. This section will detail each attempt and the associated cost estimation for each method.

4.11.1 Google Cloud Run

We began by estimating the deployment costs using the Google Cloud pricing calculator. The calculator helps us to forecast the expenses based on various parameters specific to our use case. Pricing for generative models is based on the amount of input and output data, such as the number of input and output characters for text models, and the number of generated images for image models.

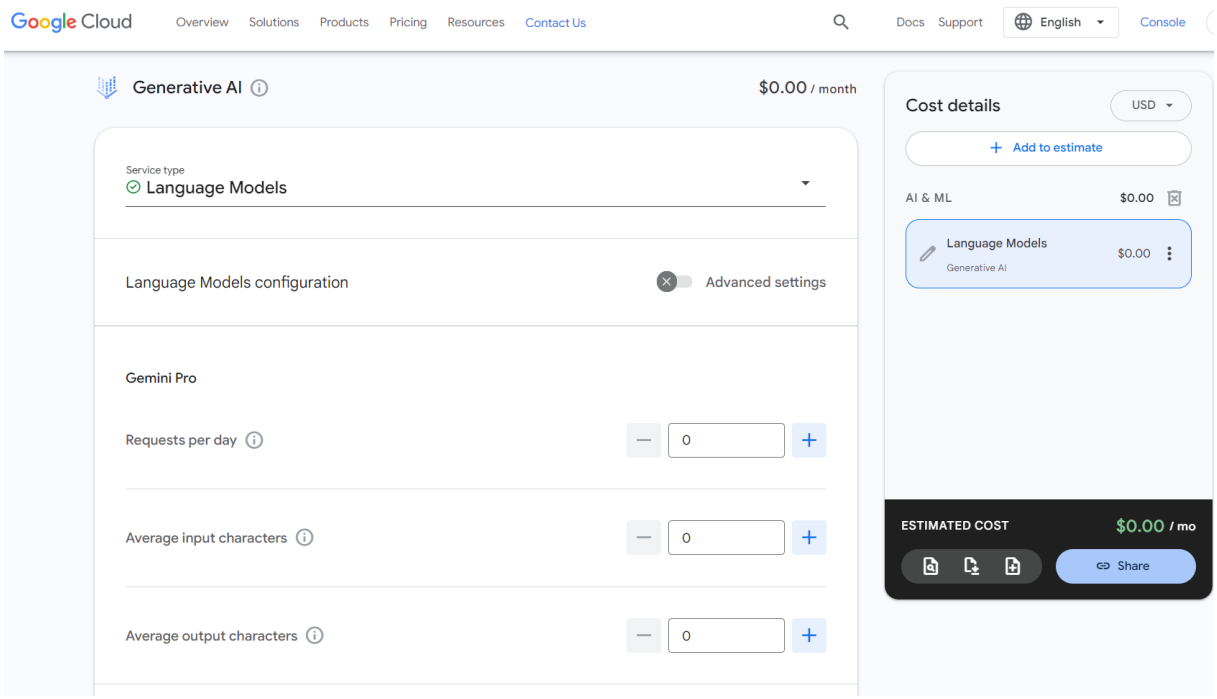


Figure 28: Google Cloud pricing calculator.

- **Requests per day:** This parameter indicates the number of requests we expect to make to the Gemini Pro model per day.
- **Average input characters:** This is the average number of characters in the input to the model.

- **Average output characters:** This parameter represents the average number of characters in the output generated by the model.

To calculate the cost estimation for Google VertexAI (using Gemini Pro), we performed a detailed analysis based on real job application demands provided by the company. This allowed us to derive accurate input and output parameters. The table below illustrates the data collected:

Seniority	Job Title	Runs	Input characters	Output characters	Average input	Average output	Max output
Junior	Java Developer	1	1667	6606	1667	6337	6938
		2	1667	5467			
		3	1667	6938			
Senior	Python Developer	1	1776	**7106**	1776	8394	10249
		2	1776	**7826**			
		3	1776	**10249**			
Junior	React Web Developer	1	1919	**6902**	1919	8822	11546
		2	1919	**11546**			
		3	1919	**8017**			
Mid	Devops Engineer	1	1920	**7936**	1920	8049	9213
		2	1920	6999			
		3	1920	9213			

Table 4: Detailed analysis of job demands showing input and output character counts across different runs and seniority levels.

Based on the results, the average input characters is 1821 and average output characters is 7900. We set the number of requests per day at 250, which led to the cost estimation:

Google Vertex AI	
Requests per day	250
Average input characters	1821
Average output characters	7900
Cost per month	22.43 €

Table 5: Cost estimation for Google Vertex AI.

Based on the data collected, we estimate the monthly cost for Google VertexAI to be approximately 22.43 euros.

Google Cloud Run allows us to deploy containerized applications that automatically scale with the number of requests.

- **CPU:** The first 180,000 vCPU-seconds are free each month. Beyond the free tier, the cost is $0.00002400 per vCPU-second.

Google Cloud Run		
CPU	Free: First 180,000 vCPU-seconds free per month	$0.00002400 / vCPU-second beyond free tier
Memory	Free: First 360,000 GiB-seconds free per month	$0.00000250 / GiB-second beyond free tier
Requests	Free: 2 million requests free per month	$0.40 / million requests beyond free tier3

Table 6: Google Cloud Run pricing details.

- **Memory:** The first 360,000 GiB-seconds are free each month. Beyond the free tier, the cost is $0.00000250 per GiB-second.
- **Requests:** The first 2 million requests are free each month. Beyond the free tier, the cost is $0.40 per million requests.

To deploy our application using Cloud Run, we packaged both applications (frontend and backend) into a single container using Docker, facilitated by a single Dockerfile. The estimated cost breakdown is shown in the following image:

- **CPU Allocation:** The CPU is allocated only during request processing, with an estimated CPU allocation time of 20,000 vCPU-seconds per month.
- **Memory Allocation:** Memory is allocated similarly, with an estimated memory allocation time of 20,000 GiB-seconds per month.
- **Requests:** We estimated 1,000,000 requests per month.

For Google Cloud Run, the cost remains within the free tier for our estimated usage, resulting in no additional cost beyond the free allocation.

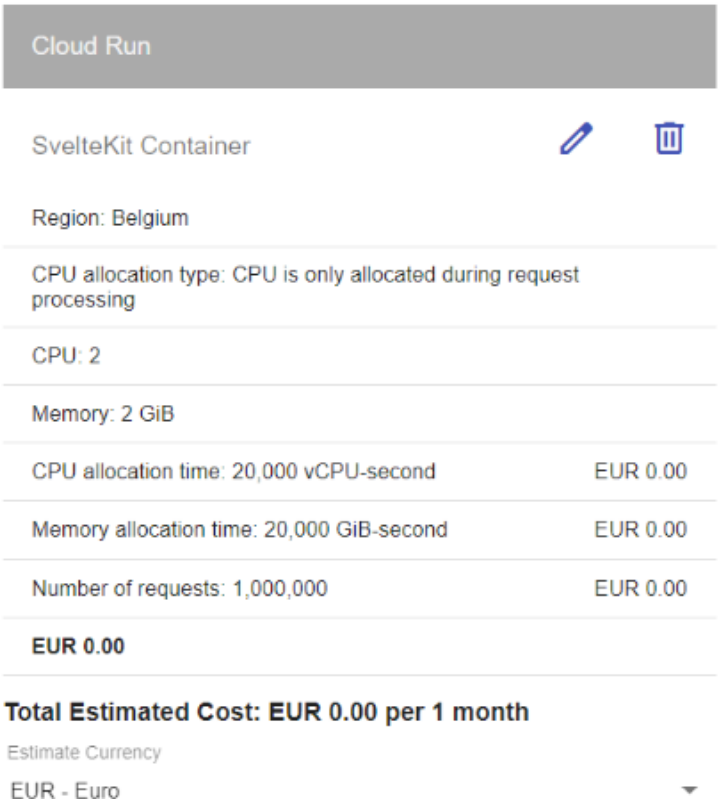


Figure 29: Estimated cost breakdown for Google Cloud Run.

4.11.1.1 Challenges Faced with Google Cloud Run Deployment

While deploying our application on Google Cloud Run seemed cost-effective, it presented several challenges that rendered it unsuitable for our needs:

- **Lack of Internal Network:** Running only a single container limits internal networking capabilities, which is crucial for our application's backend and frontend communication.
- **Cost and Complexity of Docker Compose:** Using Docker Compose with two containers and an internal bridge network increases both cost and complexity. The advantage of using two containers is that it allows the backend and frontend to communicate securely and efficiently within an isolated network, but it adds to the deployment complexity and operational costs.
- **Frontend and Backend Exposure:** The frontend must always be exposed to the backend, necessitating secure, continuous internet exposure, adding complexity and potential security concerns.
- **Authentication Issues:** Cloud Run requires requests to be authenticated. Since containers run with private IPs, requests must include an additional header parameter with a token, complicating

the setup and daily operations. This setup isn't ideal for public-facing websites due to the additional steps and potential friction it introduces for users accessing the site.

Therefore, despite its potential, Cloud Run did not meet our specific deployment requirements effectively.

4.11.2 Google App Engine

As the deployment with Google Cloud Run proved unsuitable, we explored using Google App Engine. However, several factors made it impractical for our needs:

- **Container Limitations:** Google App Engine does not support Docker images, meaning we couldn't deploy our containerized application directly. This constraint prevents leveraging container benefits such as consistent environments and isolated dependencies. Docker images were essential for us because they ensure the application runs in the same environment regardless of where it is deployed. This consistency helps avoid issues related to differences in development, testing, and production environments.
- **Project Structure:** Deploying with App Engine requires a distinct YAML-based configuration specifying the Python version, requirements, and entry point commands. This approach mandates separate deployments for backend and frontend components, complicating deployment and maintenance.
- **File System Restrictions:** App Engine does not allow file system writes, which is a significant limitation for our application, particularly for generating and saving PDFs. This restriction disrupts our workflow, where writing to the file system is essential.
- **Task Execution:** App Engine does not support threading or long-running tasks. Given our application's need to generate quizzes, this limitation poses a risk of failures or performance issues during quiz generation processes.

Due to these limitations, we concluded that Google App Engine was not a viable option for our deployment strategy.

4.11.3 Google Kubernetes Engine

After the unsuccessful attempts with Google Cloud Run and Google App Engine, we explored deploying our application using Google Kubernetes Engine (GKE). Kubernetes is a robust platform for container

orchestration, suitable for managing distributed workloads and enabling horizontal scaling of services. This approach offered advanced configuration and automation capabilities necessary for our complex application.

We changed our approach from using one container with the entire application to using two containers: one for the frontend and one for the backend. Using two containers provides several advantages:

- **Isolation of Components:** Separating the frontend and backend into different containers ensures that each component runs in its own isolated environment. This makes it easier to manage dependencies and reduces the risk of conflicts between different parts of the application.
- **Scalability:** Kubernetes enables horizontal scaling, automatically adjusting the number of container instances based on demand. When faced with increased traffic or workload, Kubernetes can dynamically spin up additional instances of the frontend or backend containers as needed. This elasticity ensures optimal performance during peak usage periods without manual intervention.
- **Improved Resource Utilization:** With independent containers for frontend and backend services, resource allocation becomes more efficient. Kubernetes allows us to allocate precise amounts of CPU and memory to each container based on its specific requirements, thereby optimizing overall resource utilization across the application.
- **Simplified Deployment and Maintenance:** Kubernetes facilitates continuous deployment practices by enabling updates to individual components (e.g., frontend updates) without disrupting the entire application. This streamlined deployment process reduces downtime and enhances operational efficiency.
- **Enhanced Security:** Containerization enhances security by isolating services. Kubernetes allows for fine-grained security policies to be applied to each container, limiting potential attack vectors and ensuring stricter access controls.

In our simulation, the estimated cost breakdown for utilizing Google Kubernetes Engine with two containers is approximately 17.31 euros. This estimation is based on a configuration with a single node using a g1-small machine type with 1 vCPU and 1.7 GB RAM, along with a 20 GiB standard persistent disk.

Deploying our application on Google Kubernetes Engine using two containers for the frontend and backend has proven to be the most effective solution. While setting up and maintaining Kubernetes can be complex and involves significant additional costs, it provides unparalleled benefits in terms of scalability, resource management, and security.

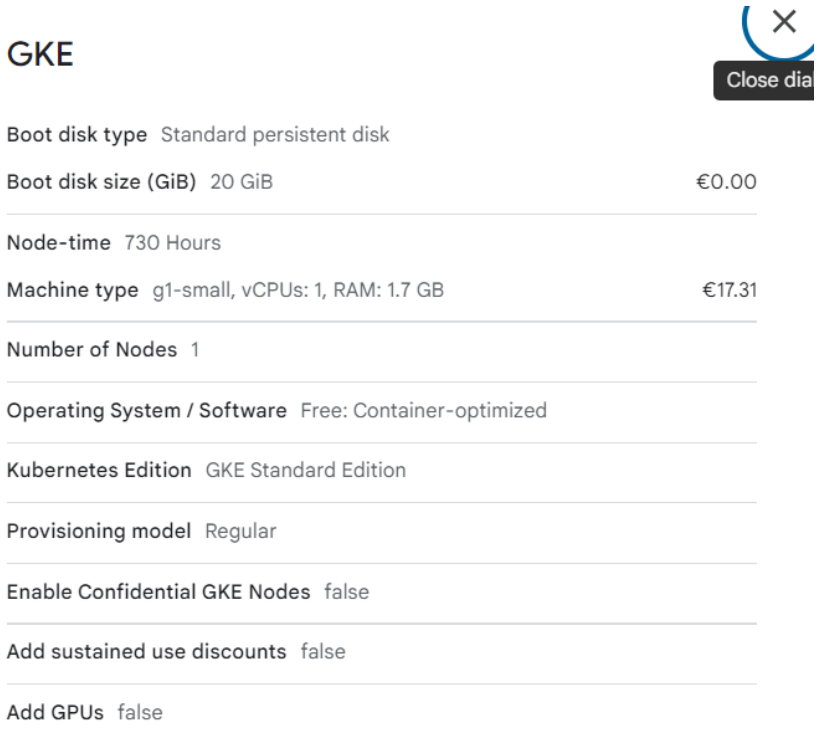


Figure 30: Estimated cost breakdown for Google Kubernetes using 2 containers.

Chapter 5

Challenges during development

During the development of our application, we encountered several challenges that required careful decision-making and problem-solving. This chapter outlines the key challenges faced and the solutions implemented to address them.

5.1 Differentiating Question Seniority Levels

The first significant issue we encountered during the prompt engineering phase involved understanding and differentiating the seniority levels of questions while using PaLM 2. Initially, we aimed to create quizzes for three seniority levels: junior, mid, and senior. However, the model struggled even to differentiate between junior and senior levels, producing similar questions for both. Due to this limitation, we decided to focus only on junior and senior levels. We invested considerable time in refining the prompts, and we faced difficulties in evaluating the questions' alignment with the specified seniority levels, necessitating stakeholder support to ensure alignment with the specified seniority levels. This challenge was subsequently mitigated with the introduction of the Gemini Pro model, which demonstrated improved capability in differentiating between the three levels.

5.2 Authentication Decisions

We had to make various decisions regarding user authentication within the application. Initially, we used temporary candidate and recruiter objects (tempCandidates and tempRecruiters) to manage user data. Although this approach worked for our Minimum Viable Product (MVP), it was not the most viable long-term solution.

- **TempCandidates:** Serve as placeholders for individuals who have yet to complete their registration. When a recruiter associates a candidate with a demand, a unique temporary identifier is

generated and added to the tempCandidates repository. This stores relevant candidate information temporarily, awaiting their official registration. Once registered, the candidate is assigned a new unique identifier, and all information is transferred to the users database. This process ensures the continuity of candidate information while transitioning from a temporary to a permanent state.

- **TempRecruiters:** Similarly, act as a registry for recruiters who have not yet completed their registration. When a recruiter initiates registration, their identifier is stored in tempRecruiters. This helps in identifying legitimate recruiters and preventing unauthorized access. Once the recruiter completes the registration, their information is moved to the users database, and the temporary entry is deleted.

The use of temporary objects (tempCandidates and tempRecruiters) before full registration is not an optimal solution for several reasons:

- **Redundancy:** It introduces unnecessary redundancy by storing user data in multiple places, first in temporary repositories and then in the main database.
- **Complexity:** This method adds complexity to the system architecture, requiring additional logic to manage the transition of user data from temporary to permanent states.
- **Resource Inefficiency:** It consumes extra computational and storage resources to handle and maintain temporary records.
- **Error-Prone:** The process increases the potential for errors, such as data inconsistency or loss during the transition.
- **Scalability Issues:** As the number of users grows, this approach may not scale efficiently, potentially leading to performance bottlenecks.

In essence, while this method was initially practical for the MVP, it is not sustainable for a robust, scalable, and efficient long-term solution.

5.3 Code Snippets Generation

When generating quizzes, the model often included code snippets within the questions. The primary issue was the format of these code snippets, as the LLMs outputs text as a string, which we convert to JSON. For this conversion to succeed, the structure must be nearly perfect.

Several issues emerge with code snippets:

1. **String Termination Issue:** Sometimes, code snippets contained string initiation characters, causing the main string to terminate prematurely, thus breaking the JSON conversion. **Example:**

```
"2. What is the correct output for this code: \n
int main(){ \n char temp[] = "Test string" ....
```

Here, the presence of " in "Test string" ends the main string unexpectedly.

Potential Solution:

- **Prompt Engineering:** Reformulate prompts to avoid using double quotes for strings in the code, instead using single quotes.
- **Limitation:** This approach might not work for languages like C and Java, where single quotes are used for characters.

2. **Line Concatenation Issue:** The model occasionally added a + character when moving to a new line, intending to concatenate strings. This caused conversion failures since Python does not handle such operations directly during JSON conversion. **Example:**

```
"2. What is the correct output for this code:" +
"          int main(){" +
"                char temp[] = "Test string" + etc
```

Here, the + characters used for concatenation disrupt the conversion process. **Potential Solution:** Eliminate the + characters, but this alone might not solve the problem since it would still result in multiple strings rather than a single continuous string.

The final output needed to be a single string in JSON format, which was often not achieved due to these formatting issues. Considering these challenges, one potential solution for both problems could involve leveraging libraries designed for escaping special characters during the conversion process. However, despite being a promising solution, there may still be limitations or specific cases where formatting issues are not automatically corrected by the library. This could necessitate exploring alternative approaches or making additional adjustments to the logic of code snippet generation. In conclusion, the generated code

snippets did not always follow a consistent format. The examples above illustrate two common issues, but the problem required careful analysis, as each format needed individual attention. Additionally, solutions that worked for one language or case might not be applicable to all. Therefore, due to these unresolved issues, we explicitly instructed the model not to generate code snippets when requesting quiz generation. This decision allowed us to avoid conversion errors and ensure the integrity of the generated quizzes.

5.4 Application Bottleneck

A specific issue was that recruiters set the time for candidates to complete quizzes, but it was difficult to determine an appropriate average response time for the quiz. A potential solution to this problem, which was not implemented, involves normalizing the response times for questions based on their difficulty. This involves several steps:

- **Define Standard Response Times:** Establish average response times for different difficulty levels (e.g., junior, mid, senior).
- **Categorize Questions by Difficulty:** Classify quiz questions into various difficulty levels, adjusting the allotted time accordingly.
- **Adjust Time Based on Difficulty:** Calculate the total quiz time by considering the difficulty of included questions, giving more time for quizzes with harder questions.
- **Continuous Monitoring and Adjustment:** Collect data on actual response times to fine-tune and adjust the standard times and difficulty categorization.
- **Statistical Analysis:** Use statistical methods to ensure the normalization process is fair and effective, making necessary adjustments based on ongoing data collection.

By implementing this solution, the application could provide a more balanced and fair assessment environment.

This approach might revisit the initial challenge of distinguishing question difficulty. However, it could be feasibly implemented through continuous monitoring and adjustment via trial and error.

5.5 Database Security

Ensuring database security was another significant challenge. Initially, the work was done with the read and write rules set to true, as security was not a primary concern in the early stages. This was necessary

because, without these settings, the backend/frontend could not access the database information. Near the end of the development, the company began receiving security warnings that any user could read and write to the database. Therefore, this issue became a priority, and we had to change the rules to tighten security using Firebase Database Rules to restrict access and ensure only authorized users could read and write data. This process required considerable time and effort to implement effective rules.

This situation exemplifies an agile practice where feedback is quickly acted upon to improve the system. In agile methodologies, teams work iteratively and adaptively, responding to issues as they arise. The prompt response to the security warnings, followed by the implementation of improved security measures, reflects the agile principle of responding to change over following a fixed plan. By prioritizing and addressing the security issue promptly, the development team demonstrated the flexibility and responsiveness that is core to agile practices.

5.6 Database Choice

Additionally, we realized that a relational database might have been more suitable for our needs. The NoSQL structure of Firebase, while flexible, made it challenging to manage growing volumes of interconnected data efficiently.

Relational databases could have provided several advantages:

- **Structured Relationships:** Relational databases excel in managing complex relationships between different entities, such as candidates, quizzes, and timestamps, through the use of tables and joins.
- **Data Integrity and Consistency:** They ensure data integrity through constraints and relationships, reducing redundancy and enhancing data accuracy.
- **Advanced Query Capabilities:** SQL allows for more advanced querying capabilities, enabling efficient retrieval and analysis of data across multiple tables.
- **Ease of Maintenance:** With structured schemas, relational databases simplify the process of maintaining and updating the database as the application evolves.
- **Indexing and Performance:** Relational databases provide robust indexing mechanisms, which can significantly improve query performance, especially as the dataset grows.

In summary, while Firebase was adequate initially, a relational database could have offered better performance, scalability, and ease of management.

5.7 Deployment Challenges

We faced multiple challenges while attempting to deploy our application. We made three attempts:

- **Google Cloud Run:** Despite its cost-effectiveness, it had technical limitations. Running a single container restricted internal networking, and using Docker Compose with two containers increased complexity and cost.
- **Google App Engine:** This was not suitable as it did not support Docker images, required a specific project structure in YAML, and had restrictions on file system writes and threading, which were essential for our application's functionality.
- **Kubernetes:** This was the final solution we considered. It offered the necessary flexibility and scalability but at a higher cost.

Chapter 6

Conclusions and future work

In this chapter, I reflect on the key achievements and lessons learned throughout the development of the recruitment application. This project successfully demonstrated the application of tools within the recruitment domain, highlighting both the potential and the challenges of integrating automated systems in this field.

6.1 Conclusions

Throughout this rigorous research endeavor, I not only successfully achieved but surpassed my proposed objectives. My journey culminated in the creation and seamless implementation of an advanced recruitment system empowered by artificial intelligence. Embracing the principles of agile methodology proved pivotal, as I navigated through numerous challenges, often requiring iterative refinement to ensure alignment with evolving project dynamics.

The mentorship and guidance of Professor Rui Castro were instrumental in steering me through intricate technical landscapes and conceptual hurdles.

Moreover, the immersive experience during my internship fostered a deep integration within the corporate environment. Collaborating closely with colleagues and mentors not only deepened my theoretical foundations but also sharpened my practical competencies, particularly in the realm of project management.

Reflecting on this journey, it is profoundly gratifying to see how we were able to architect and develop an application from inception to fruition. The culmination of this endeavor underscores the transformative power of dedication, innovation, and collaborative spirit in realizing ambitious technological solutions.

6.2 Prospect for future work

Future work on the recruitment application can focus on several promising directions:

- **CVs Screening:** One significant enhancement would be the integration of automated CVs screening capabilities. This feature, currently under development by another team in the company, aims to read and analyze CVs, extracting relevant information to assist in candidate evaluation. Implementing this functionality in our application would streamline the recruitment process further, providing recruiters with valuable insights and reducing manual effort.

- **Anti-Cheat Measures:** To enhance the integrity of the quizzes, implementing anti-cheat mechanisms would be highly beneficial. A potential solution could involve enforcing full-screen mode during quizzes to prevent candidates from switching tabs or accessing external resources. Additionally, restricting the ability to copy and paste from the quiz pages can further deter cheating. This could be implemented by disabling right-click and keyboard shortcuts for copy and paste functions during the quiz session.

- **Front-End Enhancements:** Currently, our application's front-end is quite basic and could benefit from a redesign to make it more visually appealing and engaging. Improving the aesthetics and functionality of the interface will make the application not only more attractive but also more enjoyable and easier to navigate for users.

By focusing on these areas, the recruitment application can continue to evolve, leveraging and innovative technologies to improve efficiency, security, and user experience. The lessons learned and the groundwork laid during this project provide a solid foundation for future enhancements and expansions.

Bibliography

Abdullah, M., Madain, A., and Jararweh, Y. (2022). Chatgpt: Fundamentals, applications and social impacts. In *Proceedings of the Ninth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 1–8, Milan, Italy.

Abiodun, O. et al. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4.

Abiodun, O. I. et al. (2019). Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access*, 7:158820–158846.

Alkaissi, H. and McFarlane, S. I. (2023). Artificial hallucinations in chatgpt: Implications in scientific writing. *Cureus*, 15.

Anil, R. et al. (2023). Palm 2 technical report. *ArXiv*, abs/2305.10403.

Bahrini, A. et al. (2023). Chatgpt: Applications, opportunities, and threats. *2023 Systems and Information Engineering Design Symposium (SIEDS)*, pages 274–279.

Bender, E. et al. (2021). On the dangers of stochastic parrots: Can language models be too big? *Proceedings of the 2021 Conference on Fairness, Accountability, and Transparency*, pages 610–623.

Bozkurt, A. and Sharma, R. (2023). Generative ai and prompt engineering: The art of whispering to let the genie out of the algorithmic world. *Asian Journal of Distance Education*, 18:i–vi.

Brown, T. B. et al. (2020). Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Buscemi, A. and Proverbio, D. (2024). Chatgpt vs gemini vs llama on multilingual sentiment analysis.

Chaudhari, A. R. and Joshi, S. D. (2021). Study of effect of agile software development methodology on software development process. In *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 1–4.

Chen, B. et al. (2023). Unleashing the potential of prompt engineering in large language models: a comprehensive review. *ArXiv*, abs/2310.14735.

Chowdhery, A. et al. (2022). Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

Cimpeanu, I.-A. (2023). The chatbots and their role in the progress of society. *Informatica Economica*, 27:61–77.

Eightfold AI (2024a). Eightfold AI: Unlocking the Full Potential of Talent with AI. https://eightfold.ai.

Eightfold AI (2024b). Eightfold Talent Management Data Sheet. https://eightfold.ai/wp-content/uploads/Eightfold_Talent_Management_data_sheet.pdf.

Evstropov, A. M. et al. (2023). Neural network architecture «transformer»: Artificial intelligence and its role in natural language processing. 97:189–192.

Findem AI (2024). Ai talent acquisition and management platform.

Firebase (2024). Choose a database: Cloud firestore or realtime database. https://firebase.google.com/docs/firestore/rtdb-vs-firestore.

Gong, X. et al. (2023). Evaluating the potential of leading large language models in reasoning biology questions. *ArXiv*, abs/2311.07582.

Goodfellow, I. J. et al. (2014). Generative adversarial networks.

Google Cloud (2024a). Learn about llms, palm models, and vertex ai. https://cloud.google.com/vertex-ai/docs/generative-ai/learn-resources.

Google Cloud (2024b). What is prompt engineering? https://cloud.google.com/discover/what-is-prompt-engineering.

Gupta, N. K. et al. (2023). Chatgpt: Exploring the capabilities and limitations of a large language model for conversational ai. In *2023 International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)*, pages 139–142, Faridabad, India.

Hayat, F. et al. (2019). The influence of agile methodology (Scrum) on software project management. In *2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 145–149.

HiredScore (2024). HiredScore. https://www.hiredscore.com.

HireVue (2024). HireVue: Where Hiring Happens. https://www.hirevue.com.

Hoole, Y. et al. (2023). Responsible ai at eightfold. Technical report, Eightfold AI Inc.

Jha, K. K. et al. (2021). A brief comparison on machine learning algorithms based on various applications: A comprehensive survey. In *2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, pages 1–5.

Jr., M. A. C. and Giffen, J. R. (2019). Recruiting in the robot age: Examining potential eeo implications in optimizing recruiting through the use of artificial intelligence. *The Computer and Internet Lawyer*, 36(10).

Katti, J. et al. (2022). University admission prediction using google vertex ai. In *2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR)*, pages 1–5.

Khan, R., Hussain, A., and Ahmad, S. (2023). Revolutionizing human resource management: The transformative impact of artificial intelligence (ai) applications. *International Journal of Social Science and Entrepreneurship (IJSSE)*, 3(4):306.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

Kumar, V. U. et al. (2021). Analysis and issues of artificial intelligence ethics in the process of recruitment. In *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*, pages 733–738.

Luo, Q. et al. (2023). Self-attention and transformers: Driving the evolution of large language models. In *2023 IEEE 6th International Conference on Electronic Information and Communication Technology (ICEICT)*, pages 401–405, Qingdao, China.

Madaminov, U. and Allaberganova, M. (2023). Firebase database usage and application technology in modern mobile applications. In *Proceedings of the IEEE International Conference on Advances in Power Electronics and Information Engineering (APEIE)*, pages 1690–1694.

McIntosh, T. R. et al. (2023). From google gemini to openai q* (q-star): A survey of reshaping the generative artificial intelligence (ai) research landscape. *ArXiv*, abs/2312.10868.

Meshram, R. (2023). The role of artificial intelligence (ai) in recruitment and selection of employees in the organisation. *Russian Law Journal*, 11.

Mhatre, A. (2023). Detecting the presence of social bias in GPT-3.5 using association tests. *Proceedings of the International Conference on Advanced Computational Techniques and Applications*, pages 1–6.

Mitsunaga, T. (2023). Heuristic analysis for security, privacy and bias of text generative ai: Ghatgpt-3.5 case as of june 2023. In *Proceedings of the International Conference on Computing and Networking (ICOCO)*, pages 301–305.

Mostafa, B. et al. (2020). Machine and deep learning approaches in genome: Review article. *Alfarama Journal of Basic & Applied Sciences*.

Naik, I. et al. (2023). Chat generative pre-trained transformer (chatgpt): Comprehending its operational structure, ai techniques, working, features and limitations. In *2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG)*, pages 1–9, Indore, India.

Nithya, T. and others. (2023). A comprehensive survey of machine learning: Advancements, applications, and challenges. In *Proceedings of the 2023 International Conference on Artificial Intelligence and Software Systems (ICAISS)*, pages 354–361.

Ohyver, M. et al. (2019). The comparison firebase realtime database and mysql database performance using wilcoxon signed-rank test. *Procedia Computer Science*, 157:396–405.

Paradox AI (2024). Conversational hiring software: Paradox.

Parida, P. R. and Sahu, S. (2023). Social recruitment: A credible alternative to traditional recruitment – a review. *International Journal of Research in Human Resource Management*, 5(2):17–21.

Pymetrics (2024). Pymetrics: Redefining Talent Success with AI. https://www.pymetrics.ai.

Sen, S. et al. (2023). Role of artificial intelligence-enabled recruitment processes in sourcing talent. In *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*, pages 1–5.

Sipahi, E. and Artantaş, E. (2022). Artificial intelligence in hrm. *Handbook of Research on Innovative Management Using AI in Industry 5.0*, pages 1–18.

Team, G. et al. (2024). Gemini: A family of highly capable multimodal models. *ArXiv*.

Teng, F. et al. (2023). A review on deep neural networks for ICD coding. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4357–4375.

Trziszka, M. (2023). Artificial intelligence in employee recruitment. *European Conference on Knowledge Management*.

Tullier, J. (2021). The use of artificial intelligence to recruit employees. *Journal of Undergraduate Research*, 18:5. Faculty Mentor: Cindi Bearden, BS, MAc, CPA. Business and Nonprofit Studies Department/Business Program.

van Esch, P. and Black, J. S. (2019). Factors that influence new generation candidates to engage with and complete digital, ai-enabled recruiting. *Business Horizons*.

Vaswani, A., Shazeer, N., et al. (2023). Attention is all you need.

Veluchamy, R. et al. (2021). Artificial intelligence within recruitment: Eliminating biases in human resource management. *Journal of Critical Reviews*, 08.

Wilson, C. L. et al. (2021). Auditing the pymetrics model generation process.

Zhu, X. X. et al. (2017). Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine*, 5(4):8–36.