Table of Contents

List of Figures

List of Abbreviations

S.No.	Abbreviation	Full Description
1	RAG	Retrieval Augmented Generation
2	LLM	Large Language Models
3	API	Application Programming Interface
4	RL	Reinforcement Learning
5	GPT-4	Generative Pre-Trained Transformer 4
6	MVC	Model-View-Controller
7	REST	Representational State Transfer
8	AI	Artificial Intelligence
9	SPA	single-page application
10	DOM	Document Object Model
11	MUI	Material User Interface
12	CMS	Content Management System
13	VPS	Virtual private server
14	SSH	Secure Shell

Abstract

In this age of technological advancement, most of the routine jobs and work are being automated to save time and resources. Advancements in Artificial Intelligence(AI) have not only contributed to automating various tasks that traditionally required human intervention but also have enhanced technological solutions. Previous research has primarily relied on the Retrieval-Augmented Generation(RAG) as it has heavily impacted contextual question-answering technology. Special methods, such as low-rank residual adaptation and iterative retrieval-generation synergy, have been widely used to improve RAG systems. The results of these systems have been evaluated with Generative Pre-Trained Transformer - 4 (GPT-4) showing human-level accuracy in the test. This project proposes the development of a service tailored for colleges, empowering them to integrate RAG chatbots capable of providing contextually relevant answers about their institutions based on the knowledge base of their college. By harnessing specialized RAG models the service ensures accurate and comprehensive responses to user inquiries.

Keywords: GPT-4, RAG, Contextual question-answering, Automation

Introduction

In today's era of technological advancement, universities are increasingly turning to innovative solutions to streamline operations and enhance user experience. At the forefront of this progress is Retrieval Augmented Generation (RAG) technology, which combines retrieval models, with Language Models (LLMs) to provide relevant information on a large scale. UniServe SAAS: University Service Solutions, combines artificial intelligence and administrative support to provide colleges with a service designed to empower them with RAG chatbots capable of delivering comprehensive responses to user queries. The service features a feedback system for users to indicate if the generated response was helpful, along with additional questions displayed to assist users in resolving their queries. Additionally, users of the service will be able to dynamically update the knowledge base on which the chatbot answers the queries of the users. This will greatly benefit students by providing a comprehensive solution to address most of their inquiries.

This report provides an in-depth exploration of the workings of the UniServe SAAS, examining its methodology, the existence of the previous works, and the discussion of the result and impact of the service on meeting the evolving needs of students, faculty, and administrators in educational institutions.

As the capabilities of UniServe SAAS are explored, it becomes clear that its integration signifies more than progress; it represents a fundamental change in how universities utilize AI to improve their services and adapt to the digital era's demands.

Literature Review

[1]The RAG approach is the new trend for contextual question answering because of the developments in Generative AI and Large Language Models. The RAG pipeline includes the output that is generated from the retrieval model and it is passed on to the LLM as the context so that a correct response to the user query can be generated. Many methods have been tried out to boost the RAG components concerning accuracy and to eliminate the hallucinations in the generation of answers. The basis for the RAG model was introduced by the researchers (Lewis et al., 2021) and thus, they became the foundation for the RAG model.

[2]Khatry et al. (2023) presented low-rank residual adaptation with a pre-trained embedding model to improve the retrieval model, which showed better task-specific retrieval than an embeddings-based baseline learned for more general purposes. Shao et al.(2023) covered a new paradigm of solving such tasks by introducing a beneficial aspect of using the output of the current model as an informative context to better information retrieval to support the higher quality of the generation process in subsequent iterations. In their master thesis, Li et al. (2022) characterized a recent RAG-based solution applied to an image captioning task.

[3]Reinforcement Learning (RL) has also been studied in the context of improving RAG. Bacciu et al.(2023) introduced an RL-driven method to train a retriever model tasked with searching for pertinent information within a vast database. This pertinent data is then passed on to an API-based LLM for answer generation. The researchers showcased how RL aids in diminishing hallucinations by reducing the amount of documents retrieved by the retriever. In another study, Asai et al.(2023) introduced Self RAG, a framework that trains an LLM to dynamically retrieve passages and generate answers while contemplating the retrieved passages and their outputs using reflection tokens.

[4]The researchers (Hackl et al. 2023) looked at feedback ratings produced by GPT-4 and evaluated their consistency across multiple text iterations, various periods, and differences in stylistic variation level. The study showed that GPT-4 generated similar ratings for repetitions of the same text when provided with a clear prompt. In their work (Liu et al. 2023), the researchers used GPT-4 to perform text summarization and dialogue generation tasks. They found that GPT-4 achieved state-of-the-art results for the automated evaluation, which had a high correlation with human evaluation.

Problem Statement

Transitioning students from schools to colleges presents a complex process as they leave their homes to reside on unfamiliar campuses. This often leaves them with numerous unanswered questions, which teachers or wardens typically address. To facilitate a smoother transition, there is a need for a question-answering bot or service. Implementing such a solution would benefit both students and teachers, eliminating the need for students to seek answers from multiple sources and streamlining the problem-solving process for teachers.

Methodology

The methodology section outlines the systematic approach taken to design, develop, and deploy the chatbot service. This project involves three main components: the front-end, REST API, and the RAG (Retrieval-Augmented Generation) model. Each component is integral to the overall functionality of the chatbot service, enabling seamless user interaction and intelligent response generation.

Front-End Development

Implementation

The front-end of the project is a **Client-Side Rendered (CSR)**, **single-page application (SPA)** developed using the **React** framework with **Vite**. The codebase is written in **TypeScript**, which provides the benefits of type-checking, making the code easier to debug, document, and maintain. For handling HTTP requests, **Axios** is employed as the primary library.

To facilitate form management, the **Formik** library is utilized, offering a streamlined approach to handling form state and validation. Client-side routing is managed using **React Router DOM**, ensuring efficient navigation within the **SPA**. User input validation is achieved with **Zod**, ensuring robust form handling.

For the user interface, **Material UI (MUI)** components are incorporated to enhance the visual appeal and user experience of the website. State

management is efficiently handled using the **Context API** provided by the React framework.

Server Development

Design and Architecture

The API adheres to **RESTful** principles and follows a versioning convention, prefixing each request with *v1/api*. The server is designed and implemented using the **Model-View-Controller (MVC)** architecture, which organizes the code into distinct components for handling data (Model), user interface (View), and application logic (Controller). The architectural design of the server is illustrated in Figure 4.2.1.1.

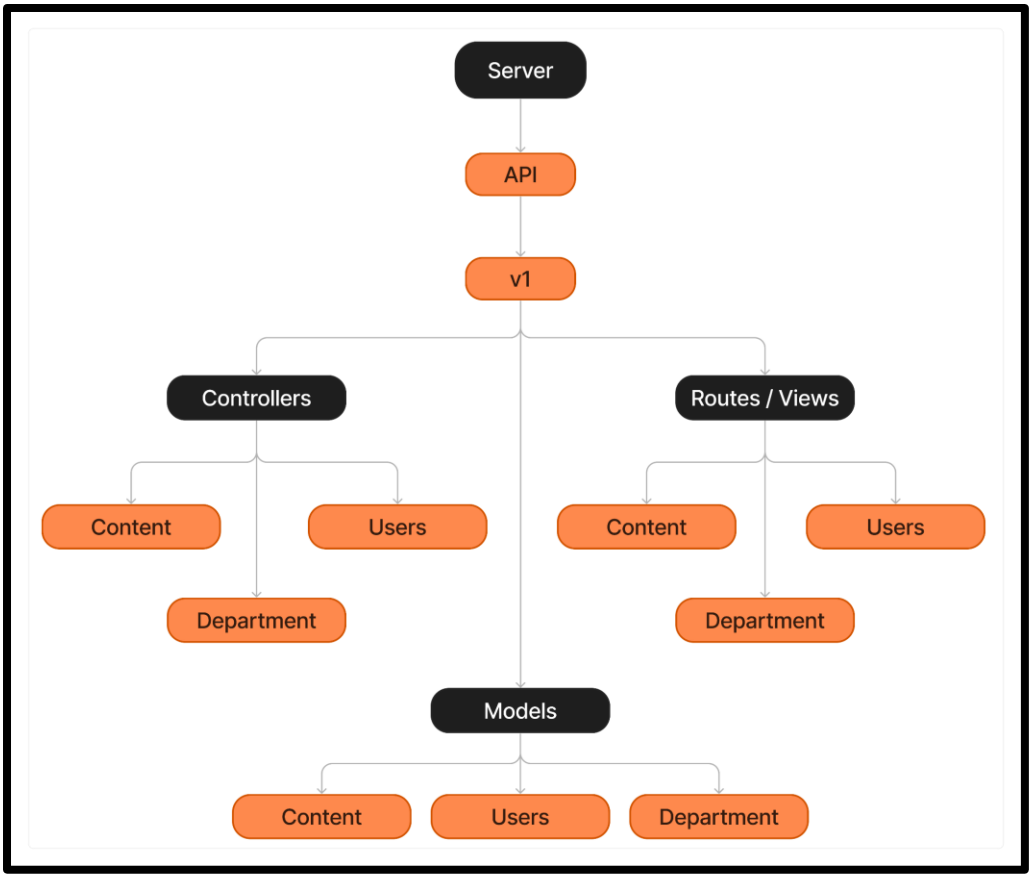

Figure 4.2.1.1 - Server's Architectural Diagram

Implementation

The server was developed using **Node.js** and written in **TypeScript**. **Express.js** was employed to manage routing and handle responses. Instead of a traditional database, the server utilizes Firebase services, including **Authentication**, **Firestore**, and **Storage**. **Zod** is used to validate incoming requests, ensuring the validity of request headers and parameters. Additionally, the server leverages a **Pinecone Client** to manage **CRUD** operations for the vector database.

RAG Model Development

Data Collection and Preprocessing

The data used to build the knowledge base for the RAG model was initially sourced from the college, containing the common inquiries received by various departments on a daily basis. Subsequently, the knowledge base was enriched with all publicly available student policies from the college website and the student handbook. As the model continued to evolve, the testing team consistently updated the data through student questionnaires and conversations with administrative staff. In the final phase of the project, a **Content Management System (CMS)** was developed to directly integrate updates from college authorities, ensuring that any policy changes or new information could be promptly reflected in the chatbot model.

While the model can accommodate different types of textual data, it favors Word documents due to their ease of extracting textual information. The data corpus, initially comprised of Word documents,

was divided into chunks and then uploaded to the Pinecone vector database using **multithreading** and the Huggingface embedding model. The integration of multithreading and the **Huggingface Embedding model "avsolatorio/GIST-Embedding-v0"** (Solatorio, 2024) resulted in a considerably faster data upload to the vector store compared to conventional methods.

Visual Presentation and Discussion of Results

The successful completion of this project has led to the development of a website tailored for use by students (users) and college administrators (customers). The student portal is depicted in Figure 5.1, while the CMS interfaces for college administrators are illustrated in Figures 5.2, 5.3, and 5.4.

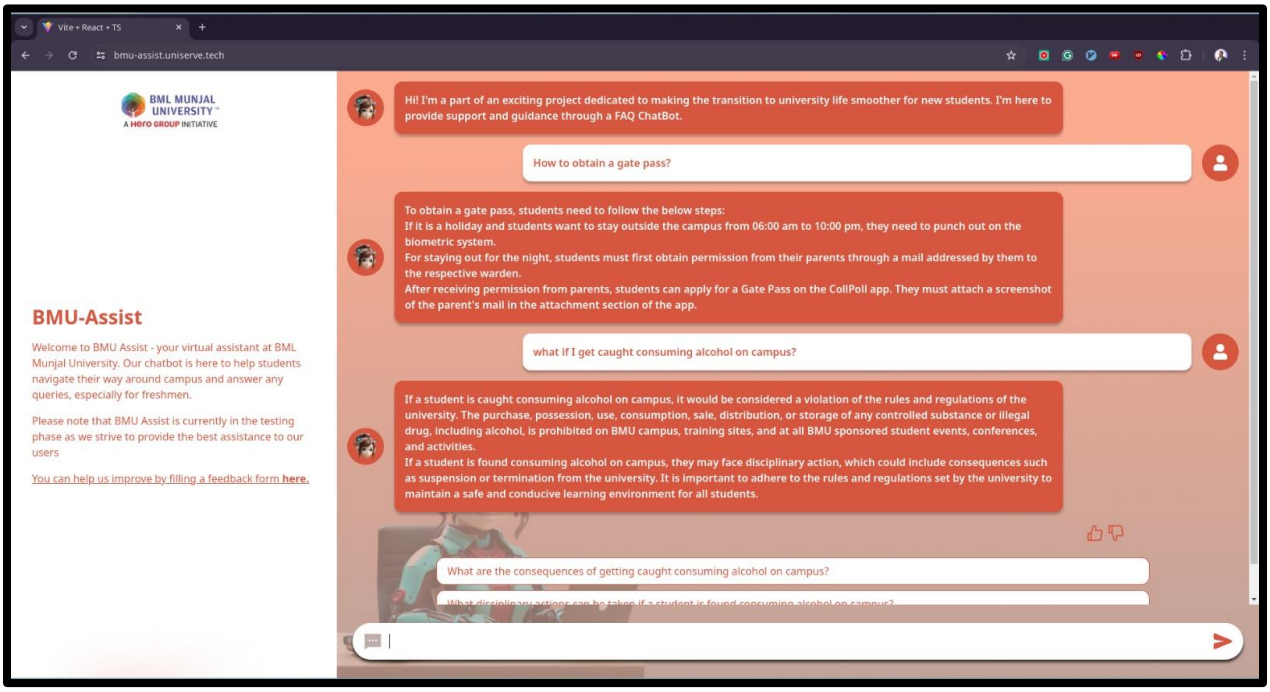

Figure 5.1 - BMU Assist Chatbot

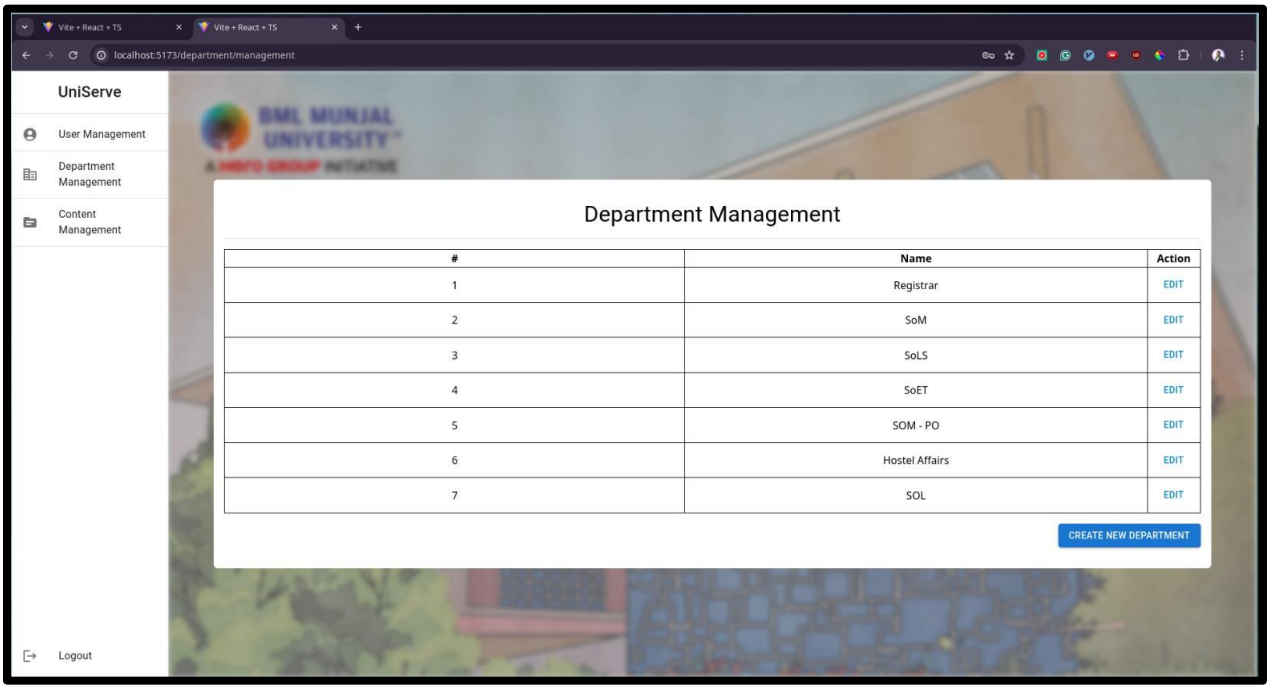

Figure 5.2 - Department Management System

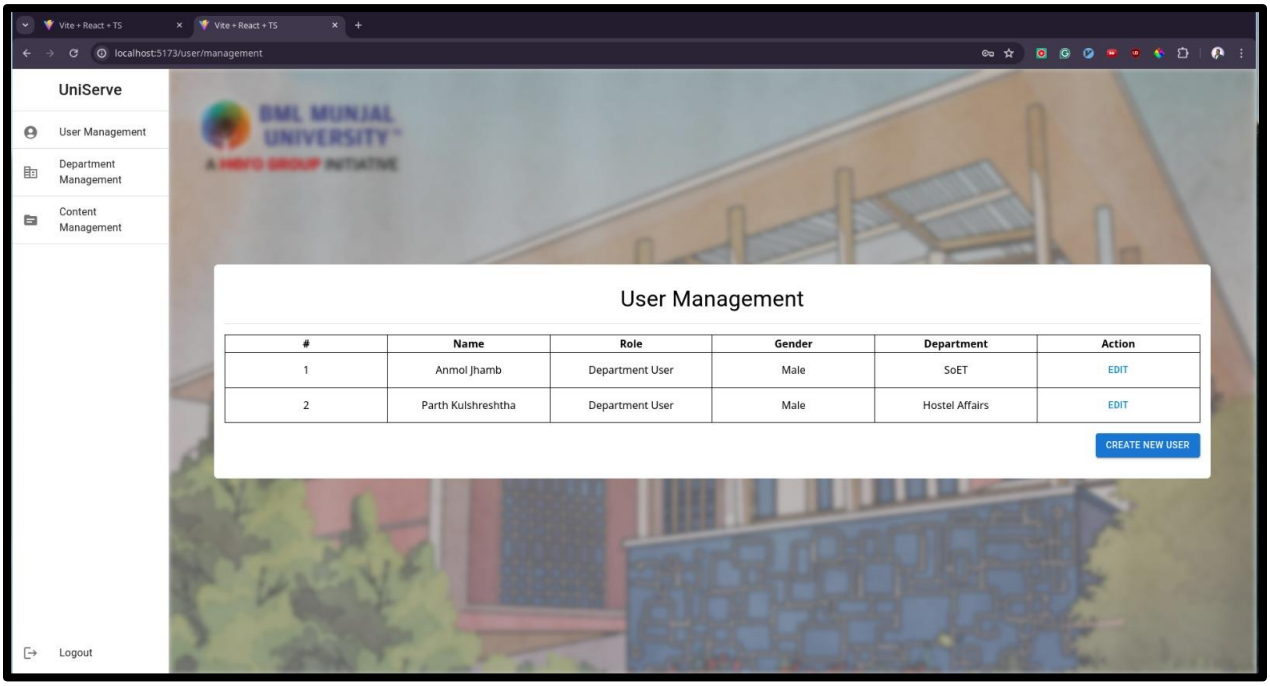

Figure 5.3 - User Management System

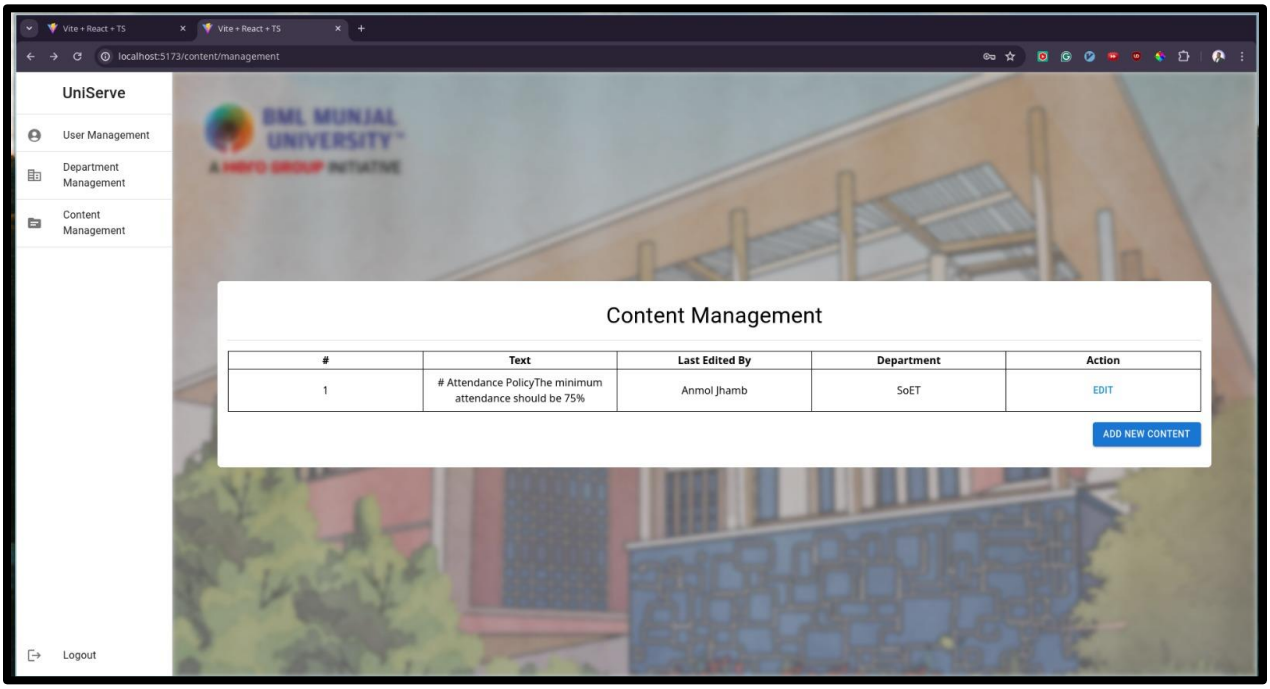

Figure 5.4 - Content Management System

Utilizing Firebase and Pinecone databases in the server, the application attains high scalability due to the "pay as you grow" model offered by both services. The front end is hosted on Vercel, which also provides similar scalability. Additionally, the model is lightweight, requiring only 2GB of RAM to manage incoming requests, and it can be easily horizontally scalable by integrating a load balancer in DigitalOcean.

The model has been undergoing continuous testing by both students and teachers since its deployment. Approximately 190 students have tested the model and provided feedback.

Future Scope

Ethical Considerations

It is crucial to take ethical considerations into account when responding. While the current ethical considerations are based on the GPT-3.5-turbo base model, additional ethical layers need to be added when the service is utilized by educational institutions to ensure that the bot's generated responses are free from discrepancies.

Expanding Accessibility: Transitioning from Website to Mobile App

Shifting our chatbot from the website version to the mobile app will result in increased user accessibility by increasing user engagement and this will take us a step further by extending the customer base that mainly relies on mobile phones for access. By taking the edge of mobile app capabilities, we can deliver a more intuitive, stable, and well-performing chatbot for our customers.

Conclusion

Based on the testing and evaluation of the generated response, it can be concluded that the BMU Assist functions effectively. The bot, utilizing its knowledge base, is proficient in addressing most student queries. Furthermore, unanswered queries are consistently updated through the bot's feedback system. Additionally, teachers and administrative staff have the capability to directly enhance the knowledge base, leading to continuous improvements in the bot's performance over time.

This service is suitable for use in any institution. With an expanded dataset, the model or bot will be better equipped to provide detailed responses to student inquiries, thereby significantly aiding students.

References

1. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2021). Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv*. https://arxiv.org/abs/2005.11401
2. Khatry et al. 2023. Augmented Embeddings for Custom Retrievals. arXiv:2310.05380.
3. Shao, Z.; Gong, Y.; Shen, Y.; Huang, M.; Duan, N.; and Chen, W. 2023. Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy. arXiv:2305.15294.
4. Li, H.; Su, Y.; Cai, D.; Wang, Y.; and Liu, L. 2022. A survey on retrieval-augmented text generation. arXiv:2202.01110.
5. Bacciu, A.; Cocunasu, F.; Siciliano, F.; Silvestri, F.; Tonellotto, N.; and Trappolini, G. 2023. RRAML: Reinforced Retrieval Augmented Machine Learning.
6. Asai, A.; Wu, Z.; Wang, Y.; Sil, A.; and Hajishirzi, H. 2023. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. arXiv:2310.11511.
7. Hackl, V.; Müller, A. E.; Granitzer, M.; and Sailer, M. 2023. Is GPT-4 a reliable rater? Evaluating Consistency in GPT-4 Text Ratings. arXiv:2308.02575.
8. Liu, Y.; Iter, D.; Xu, Y.; Wang, S.; Xu, R.; and Zhu, C. 2023b. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. arXiv:2303.16634.

9. Solatorio, A.V. (2024). GISTEmbed: Guided In-sample Selection of Training Negatives for Text Embedding Fine-tuning. *arXiv preprint arXiv:2402.16829*. Retrieved from https://arxiv.org/abs/2402.16829
10. Google. (n.d.). Listen to real-time updates with Firestore. Firebase. Retrieved May 16, 2024, from https://firebase.google.com/docs/firestore/query-data/listen