**VIVEKANAND EDUCATION SOCIETY'S**
**INSTITUTE OF TECHNOLOGY**

**Department of Computer Engineering**


Project Report on

# RAG for Railways

In partial fulfillment of the Fourth Year (Semester–VII), Bachelor of Engineering (B.E.) Degree in Computer Engineering at the University of Mumbai

Academic Year 2024-2025

**Dr. Mrs.Gresha Bhatia**

**Submitted by**

Aryan Raje , D17A/51
Arya Raje, D17A/50
Ishita Marathe, D17A/41
Prasad Lahane, D17A/36

(2024-25)

**VIVEKANAND EDUCATION SOCIETY'S**
**INSTITUTE OF TECHNOLOGY**

## Department of Computer Engineering


# CERTIFICATE of Approval

This is to certify that ***Aryan Raje(D17A-51), Arya Raje(D17A-50), Ishita Marathe(D17A-41), Prasad Lahane(D17A-36)***of Fourth Year Computer Engineering studying under the University of Mumbai has satisfactorily presented the project on "***RAG for Railways***" as a part of the coursework of PROJECT-I for Semester-VII under the guidance of ***Dr. Mrs.Gresha Bhatia*** in the year 2024-2025.

_____

Date

_____ _____

Internal Examiner External Examiner

_____ _____ _____

Project Mentor Head of the Department Principal

Dr. Mrs. Nupur Giri Dr. J. M. Nair

# ACKNOWLEDGEMENT

We are thankful to our college Vivekanand Education Society's Institute of Technology for considering our project and extending help at all stages needed during our work of collecting information regarding the project.

It gives us immense pleasure to express our deep and sincere gratitude to Deputy Head of Department **Dr. Mrs.Gresha Bhatia** (Project Guide) for her kind help and valuable advice during the development of project synopsis and for her guidance and suggestions.

We are deeply indebted to Head of the Computer Department **Dr.(Mrs.) Nupur Giri** and our Principal **Dr. (Mrs.) J.M. Nair ,** for giving us this valuable opportunity to do this project.

We express our hearty thanks to them for their assistance without which it would have been difficult in finishing this project synopsis and project review successfully.

We convey our deep sense of gratitude to all teaching and non-teaching staff for their constant encouragement, support and selfless help throughout the project work. It is a great pleasure to acknowledge the help and suggestion, which we received from the Department of Computer Engineering.

We wish to express our profound thanks to all those who helped us in gathering information about the project. Our families too have provided moral support and encouragement several times.

# Computer Engineering Department

## COURSE OUTCOMES FOR B.E PROJECT

Learners will be to:-

| Course Outcome | Description of the Course Outcome |
|---|---|
| CO 1 | Do literature survey/industrial visit and identify the problem of the selected project topic. |
| CO2 | Apply basic engineering fundamental in the domain of practical applications FORproblem identification, formulation and solution |
| CO 3 | Attempt & Design a problem solution in a right approach to complex problems |
| CO 4 | Cultivate the habit of working in a team |
| CO 5 | Correlate the theoretical and experimental/simulations results and draw the proper inferences |
| CO 6 | Demonstrate the knowledge, skills and attitudes of a professional engineer & Prepare report as per the standard guidelines. |

# ABSTRACT

This project, titled "Retrieval-Augmented Generation (RAG) for Railways," explores the application of RAG technology to enhance railway information systems, specifically for Konkan Railways. RAG combines information retrieval techniques with large language models (LLMs) to generate accurate and contextually relevant responses to natural language queries. This project addresses the limitations of traditional railway apps, which often require users to manually navigate complex menus and schedules. By leveraging natural language processing (NLP), users can query the system conversationally and receive real-time train schedules and route information. The system is built using an external knowledge base of railway data, vectorized through NLP techniques, enabling efficient search and retrieval. The augmented data then powers the LLM to generate precise responses. This project holds significant technological and social relevance, providing a user-friendly interface for both regular commuters and tourists, and promoting seamless travel experiences.

# INDEX

# Chapter 1 : Introduction

## 1.1. Introduction to the project

Retrieval-Augmented Generation (RAG) is an innovative architectural approach designed to enhance the effectiveness of large language model (LLM) applications by utilizing specialized data. This technique involves retrieving relevant documents or data pertinent to a specific query or task, which then serves as context for the language model. RAG has proven particularly effective in applications such as chatbots and question-and-answer systems that require access to up-to-date information or domain-specific knowledge.

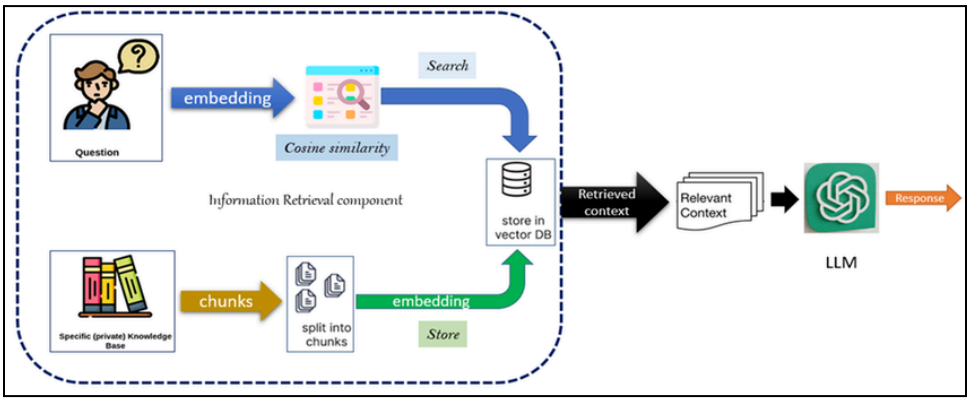



*Fig.1. RAG system*

*(Source : Semantic Embeddings for Arabic Retrieval Augmented Generation (ARAG), January 2023, International Journal of Advanced Computer Science and Applications 14, DOI:10.14569/IJACSA.2023.01411135, License CC BY 4.0)*

In the contemporary railway industry, effective management and analysis of vast datasets—including operational data, maintenance logs, safety reports, and customer feedback—are crucial for ensuring seamless operations and delivering high-quality services. As the complexity and volume of this data continue to grow, traditional data analysis methods often fall short in providing timely and actionable insights, which can hinder decision-making processes and impact service delivery.

To tackle these challenges, our project, titled "RAG for Railways," explores the application of RAG within the railway sector. By combining retrieval techniques with generative language models, RAG aims to enhance both information retrieval and content generation, ultimately improving the accuracy, relevance, and comprehensiveness of reports and responses related to

railway operations. In today's landscape, chatbots like ChatGPT exemplify the capabilities of large language models in addressing complex queries. Similarly, we propose the development of domain-specific chatbots that enable users to ask questions related to railway services and receive precise, contextually relevant answers. Current systems, like M-indicator, provide useful information on railway schedules but require users to navigate through extensive menus manually. These systems do not support natural language queries, leading to a less efficient user experience.


*Fig.2.Logo Konkan Railway*
*(Source:Konkan Railway.com , irctc.co.in)*

Our initiative seeks to revolutionize how users access railway information by facilitating natural language queries. Users will be able to pose questions in English—such as inquiries about train timings or routes—and receive prompt and accurate answers. By narrowing the project's focus to Konkan Railways and its scheduling information, we aim to create a tailored solution that meets the specific needs of this regional service. In summary, Retrieval-Augmented Generation represents a significant advancement in how information is retrieved and generated. By leveraging both the retrieval of relevant data and the generative capabilities of language models, RAG empowers users to ask questions in natural language and receive grounded, accurate responses. This makes RAG especially beneficial for specialized applications like the railway sector, where timely and precise information is critical. Through this project, we aim to improve user experience and operational efficiency within the railway industry, transforming how users interact with and access vital information.

## 1.2. Motivation for the project

The motivation for this project stems from the challenges many users face when interacting with traditional railway apps. These applications often require users to navigate through complex charts and schedules, which can be cumbersome and frustrating. By allowing users to type queries in natural language, RAG provides a more intuitive and efficient way to obtain information, eliminating the need for tedious scrolling. Additionally, our focus on Konkan Railways is driven by the issues of late and untimely trains, which highlight the need for better information access. One member of our team has established contacts with railway officials, facilitating data collection and enhancing the project's efficiency. This combination of user-friendly technology and direct access to relevant data positions us to make a meaningful impact.

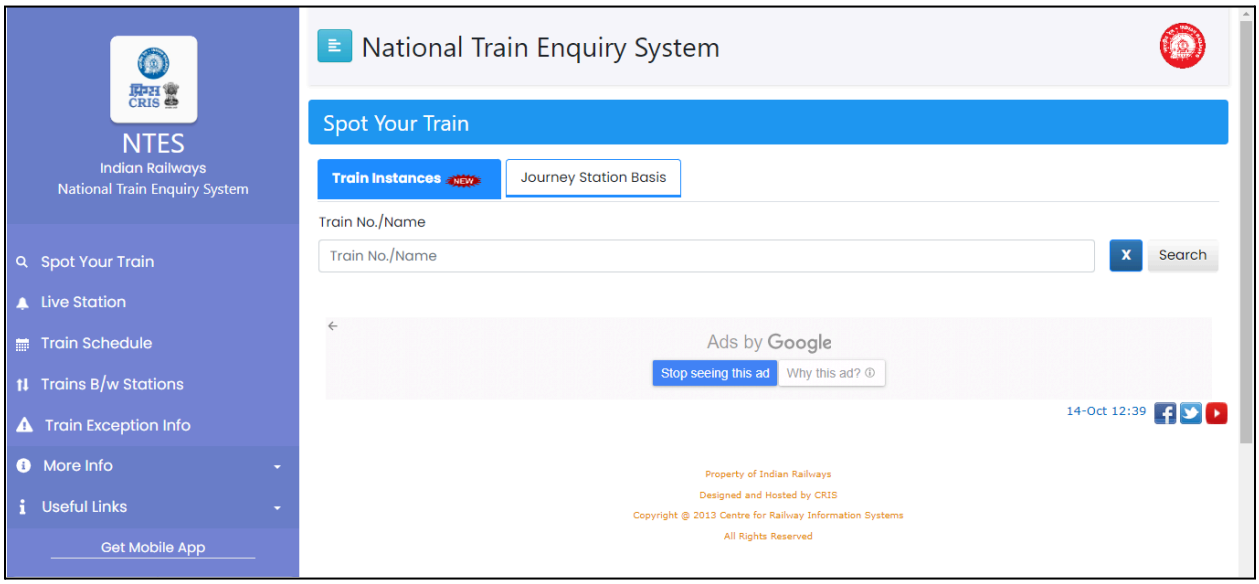



*Fig.3.NTES dashboard*

*(Source:https://enquiry.indianrail.gov.in/mntes/)*

## 1.3. Drawback of the existing system

1. Context Management for Long Documents: Existing systems struggle with efficiently managing long contexts, which could be crucial for handling detailed train schedules, routes, and real-time updates.[1][7]
2. Robustness to Misinformation: RAG system assumes correctness of documents provided.[4][9]
3. Integration of Domain-Specific Knowledge: Current RAG systems are not be well-adapted to the specific needs of specific information and integrating with live data sources.[1][7]
4. Resource Intensive Operations: High computational cost and memory requirements impacts the feasibility of deploying a RAG system at scale for real-time updates and queries.[1][8][10]
5. Generalization and Domain Adaptation: Existing RAG methods does not perform optimally across different domains, which affects the system's ability to handle the varied queries and requirements of users effectively.[8][11]
6. Querying in natural language: No such system is built that queries a railway system in natural i.e english language.

## 1.4. Problem Definition

Many existing railway information systems and apps rely on **manual selection** of stations, train numbers, and routes, which can make them less user-friendly, especially for users unfamiliar with specific railway details. This approach poses a challenge to both tourists and occasional travelers who might not have knowledge of the exact train number, station codes, or the intricacies of train routes. Navigating through complex menus and inputting precise details can become a frustrating experience for these users.

**Allowing users to interact with the system using natural language queries would significantly enhance the overall experience, enabling them to easily ask questions and quickly retrieve relevant information without needing to navigate through complex menus or know specific train details**.

Using an extensive knowledge base, our system will aim to deliver accurate and relevant information in response to these queries, making it easier for users to access and understand the needed details.

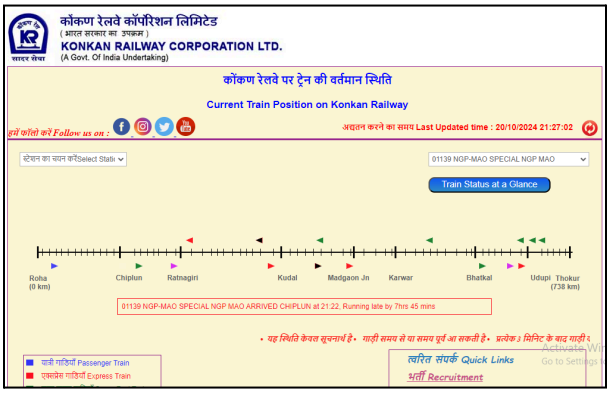



*Fig. 4.1. Current Train Position on Konkan Railway*
*(Source: https://konkanrailway.com/VisualTrain/)*

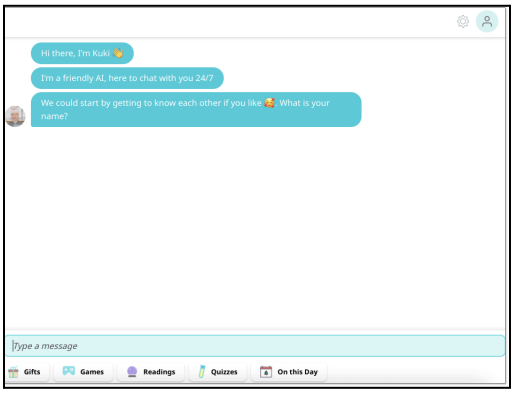



*Fig. 4.2. Chatbot Interface*
*(Source:https://helpcrunch.com/blog/chatbot-ui/)*

The focus of this project is on the **Konkan Railway** system, which is renowned for its scenic routes and complex schedule network, especially along the west coast of India. This makes it an ideal use case for our RAG system, given the diversity of users (including tourists) and the challenge of navigating schedules for different stations.

## 1.5 Relevance of the Project

The **RAG for Railways** project is highly relevant in today's context as it addresses key challenges faced by users when interacting with traditional railway information systems. As India's railway network expands and the number of travelers increases, there is a growing need for more **intuitive and accessible interfaces**. Conventional railway apps often require users to input specific train numbers or station codes, making them less user-friendly for individuals unfamiliar with the routes or local terminologies, particularly tourists and occasional travelers.

With the Konkan Railway system being a major transportation route in western India, particularly for tourists visiting the coastal areas, this project's focus on simplifying access to schedule information is both timely and critical. By leveraging **natural language processing (NLP)** and **Retrieval Augmented Generation (RAG)**, we enable users to query the system in a more conversational manner, improving overall usability and enhancing the travel experience.

This project is also highly relevant from a **technological innovation** perspective. The integration of RAG into a real-time railway system not only showcases the practical application of cutting-edge artificial intelligence but also sets a precedent for how public transportation systems can adopt AI-driven solutions. As we move towards more **smart transportation systems**, this project acts as a stepping stone towards building more efficient, AI-powered public services.

Moreover, by targeting the **Konkan Railway**, known for its scenic but complex routes, the project tackles an important logistical challenge. Ensuring easy access to train schedules in such areas is crucial for promoting tourism, improving travel planning, and providing an inclusive solution that caters to a wide range of users, from locals to international travelers. In this regard, the project holds substantial **social relevance** as well, contributing to a more connected and informed public transport system.

## 1.6 Methodology used

The **RAG for Railways** system is built using a structured methodology to ensure efficient and accurate information retrieval. The following steps outline the key components of the system's architecture:

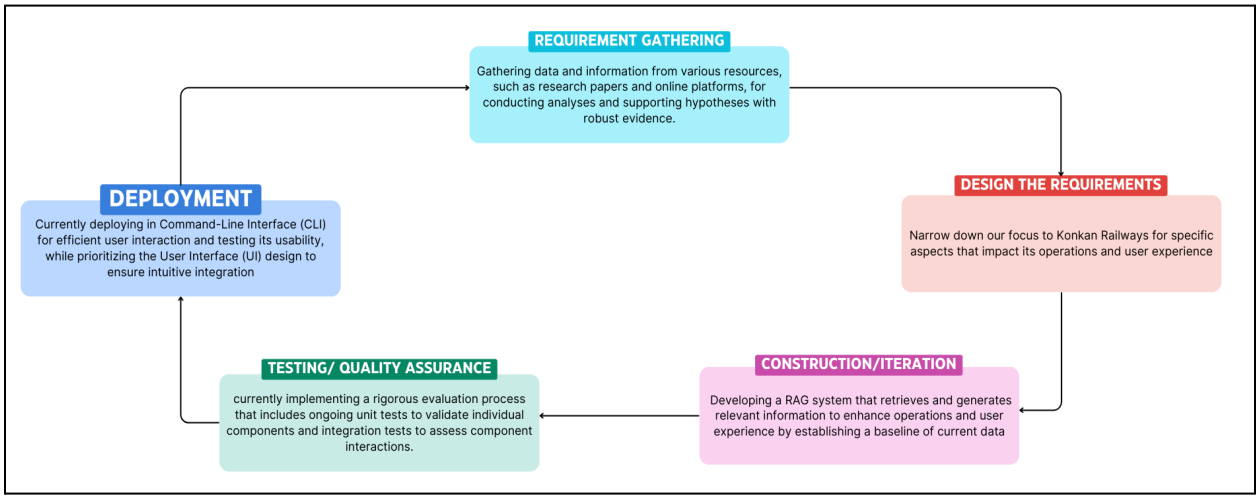



*Fig.5.Agile Methodology*

**Requirement Gathering**

Analyzing the existing systems, Evaluating the lacuna,Identifying the end users,Identifying the system.

Analyzing Existing Systems: We begin by conducting a thorough analysis of the current systems to understand their functionalities and limitations.

Evaluating the Lacunae: Next, we assess the gaps and shortcomings within these systems that hinder their effectiveness.

Identifying End Users: It's essential to identify the end users who will interact with the system, ensuring that their needs and preferences are considered.

Identifying the System: Finally, we define the specific system requirements and parameters to guide the development of an improved solution.

**Design the requirements:**

Create External Data Source

1. Data Collection: The system relies on a comprehensive external data source, which includes railway schedules, station information, routes, and other relevant details specific to the Konkan Railway.
2. Data Structuring: This data is processed and structured in a way that it can be easily accessed and searched, ensuring it is well-organized for efficient retrieval.

**Construction:**

Convert Data to Vectors

1. Vectorization: Once the data is collected, it is converted into vector representations using natural language processing (NLP) techniques. This transformation allows the system to better understand the semantics of the data.
2. Vector Database Storage: These vectorized representations are stored in a vector database, enabling fast and accurate search and retrieval.

**Testing and Deployment:**

Retrieve Relevant Information

1. User Query Vectorization: When a user submits a query, the system converts the query into a vector representation using the same NLP model that was used for the external data.
2. Search Database: The vectorized query is then compared with the vectors in the database to identify the most relevant information.
3. Document Retrieval: Based on similarity scores, the system retrieves the most relevant documents or schedule information from the vector database.

Augment the LLM Prompt

1. Augmenting the LLM Prompt: After retrieving the relevant documents, the information is used to augment the LLM (Large Language Model) prompt. The retrieved data is combined with the user's query to provide context for the LLM.
2. Generate Response: The LLM processes the augmented prompt and generates a coherent, accurate response, providing the user with the required railway schedule or other related information.

Update External Data

1. Handling Stale Data: To ensure the system remains accurate over time, automated updates or batch processing methods are used to periodically refresh the external data source.
2. Current Data: This approach ensures that the data within the vector database remains up-to-date, eliminating issues caused by outdated or stale information, and maintaining high-quality responses for users

# Chapter 2: Literature Survey

## 2.1. Research Papers

We have conducted a comprehensive analysis of various research papers to explore the latest advancements and methodologies in this area. Also, we went through an article regarding the railway infrastructure. This review highlights key findings and contributions from prominent studies, shedding light on the effectiveness of RAG in enhancing language models. Below are some of the significant papers that have informed my research and provided valuable insights into the topic.

### 2.1.1 Abstract of the research paper

**Paper 1:**

An analysis of performance of Indian railways[4] Hafiz Wasim Akram et.al

*Int. J. Logistics Systems and Management, Vol. 40, No. 3, 2021*

*20 January, 2022*

***Abstract:***

Indian Railways (IR) has experienced a significant decline in both logistical and financial performance, with issues such as poor Operating Ratio (OR), Cost of Revenue (COR), and Return on Logistics Assets Ratio (ROLAR)**.** Current literature on IR's performance lacks adequate diagnosis of these problems, necessitating deeper analysis.

**Paper 2:**

Seven Failure Points When Engineering a Retrieval Augmented Generation System[7]

Scott Barnett et.al

*2024 IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN)*

*18 June, 2024*

***Abstract:***

This paper presents an experience report on the failure points of RAG systems from three case studies from separate domains: research, education, and biomedical.

**Paper 3:**

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks[1]

Patrick Lewis et.al

*34th Conference on Neural Information Processing Systems(NeurIPS), Vancouver, Canada, 22 May, 2020*

***Abstract:***

Introduction to RAG-models where parametric-memory is pre-trained seq2seq model. The non-parametric memory is a dense Wikipedia vector index, accessed via a pre-trained retriever.

**Paper 4:**

Fine-Tune The Entire Rag Architecture (Including Dpr Retriever)For Question-Answering[8]

Shamane Siriwardhana et.al

*arXiv:2106.11517v1*

*23 June, 2021*

***Abstract:***

Fine-tune the entire Retrieval Augment Generation (RAG) architecture in an end-to-end manner and highlighted main engineering challenges that needed to be addressed to achieve this objective.

**Paper 5:**

PaperQA: Retrieval-Augmented Generative Agent forScientific Research[9] Jakub Lála et.al

*arXiv:2312.07559v2 14 December, 2023*

***Abstract:***

Viewing agent as a question-answering model, this paper finds it exceeds performance of existing LLMs and LLM agents on current science QA benchmarks.

**Paper 6:**

Retrieval-Augmented Generation for Large Language Models: A Survey[10]

Yunfan Gao et.al *arXiv:2312.10997v5 [cs.CL]*

*27 March, 2024*

***Abstract:***

The paper highlights the state-of-the art technologies embedded in each of these critical RAG components, providing a profound understanding of the advancements in RAG systems. Furthermore, this paper introduces an up-to-date evaluation framework and benchmark

**Paper 7:**

FlashRAG: Modular Toolkit for Efficient Retrieval-Augmented Generation Research[11]

Jiajie Jin et.al *arXiv:2405.13576v1 [cs.CL]*

*22 May, 2024*

***Abstract:***

Toolkit implements 12 advanced RAG methods and has gathered and organized 32 benchmark datasets. Toolkit has various features, including customizable modular framework, rich collection of pre-implemented RAG works, comprehensive datasets, efficient auxiliary preprocessing scripts, and extensive and standard evaluation metrics

**Paper 8:**

Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation[16]

Satyapriya Krishna et.al *arXiv:2409.12941 [cs.CL]*

*19 Sept, 2024*

***Abstract:***

FRAMES is a comprehensive evaluation dataset designed to test LLMs' ability to provide factual, retrieval-augmented, and reasoned responses to multi-hop questions, demonstrating significant performance improvements with a multi-step retrieval pipeline.

**Paper 9:**

Boosting Healthcare LLMs Through Retrieved Context[15]

Jordi Bayarri-Planas, et.al *arXiv:2409.13127v1 [cs.AI]*

*23 Sept, 2024*

***Abstract:***

This study examines the enhancement of LLM factuality in healthcare through optimized context retrieval methods, proposing the OpenMedPrompt pipeline, which improves open-ended answer generation and narrows the performance gap between open LLMs and private solutions in medical benchmarks.

**Paper 10:**

Enhancing Structured-Data Retrieval with GraphRAG: Soccer Data Case Study[14]

Zahra Sepasdar et.al *arXiv:2409.17580v1 [cs.IR]*

*26 Sept, 2024*

***Abstract:***

Structured-GraphRAG is a versatile framework that enhances information retrieval from structured datasets by using knowledge graphs to improve accuracy, efficiency, and reliability in natural language queries, outperforming traditional retrieval-augmented generation methods.

**Paper 11:**

MemoRAG: Moving Towards Next-Gen RAG via Memory-Inspired Knowledge Discovery[13]

Hongjin Qian et.al *arXiv:2409.055916v2 [cs.CL]*

*10 Sept, 2024*

***Abstract:***

MemoRAG is a novel retrieval-augmented generation framework with long-term memory, utilizing a dual-system architecture to improve task performance by generating draft answers that guide retrieval and then producing final answers with a more expressive LLM, outperforming traditional RAG systems on complex and straightforward tasks.

**Paper 12:**

Agentic Retrieval-Augmented Generation for Time Series Analysis[12]

Chidaksh Ravuru et.al *arXiv:2408.144846v1 [cs.CL]*

*18 Aug, 2024*

***Abstract:***

This novel agentic RAG framework for time series analysis uses a hierarchical multi-agent architecture with specialized sub-agents, leveraging fine-tuned language models and a shared prompt repository to improve predictions and outperform traditional task-specific methods on major time series tasks.

### 2.1.2 Inferences drawn from the paper

1. **Indian Railways performance analysis**: The study highlights the logistical and financial decline of Indian Railways, emphasizing problems like a poor Operating Ratio (OR), Cost of Revenue (COR), and Return on Logistics Assets Ratio (ROLAR), signaling a need for deeper analysis of these issues [4].
2. **Seven Failure Points in RAG Systems**: This paper identifies common failure points when engineering RAG systems in diverse domains (research, education, and biomedical), stressing the challenges and necessary refinements for robust system performance [7].
3. **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**: This research introduces the RAG model, utilizing pre-trained seq2seq models (parametric memory) and dense vector indexing of Wikipedia (non-parametric memory) for knowledge-intensive NLP tasks [1].
4. **Fine-Tuning RAG Architecture for Question Answering**: The paper focuses on fine-tuning RAG architectures, particularly the DPR retriever, while addressing the engineering complexities that arise in end-to-end tuning for effective question-answering [8].
5. **PaperQA: A RAG-based Agent for Scientific Research**: This research demonstrates the superior performance of RAG-based agents over existing LLMs in science-based QA benchmarks, underscoring its efficacy in scientific research applications [9].
6. **Survey of RAG for Large Language Models**: The paper offers a comprehensive review of the advancements in RAG systems, exploring critical components, evaluation frameworks, and cutting-edge technologies [10].
7. **FlashRAG Toolkit**: This paper introduces the FlashRAG toolkit, which supports modular and efficient RAG research through 12 advanced RAG methods, a rich collection of datasets, and comprehensive evaluation metrics [11].

## 2.2. Articles referred

We referred to this article that provided us with more insights into the railway infrastructure.

**Evaluating the impact of infrastructure development: Case Study of Konkan Railway in India**

*International initiative for impact innovation, Impact Evaluation Report 114, March 2020*

*Sreeja Jaiswal, Gunther Bensch, Aniket Navalkar, T Jayaraman, Kamal Murari, Unmesh Patnaik.*

The article is a final report on an impact evaluation funded by the International Initiative for Impact Evaluation (3ie) titled "Evaluating the impact of infrastructure development: case study of the Konkan Railway in India." The report assesses the social and economic outcomes of the Konkan Railway, a major infrastructure project in India, to understand its impact on development in the region. The evaluation is part of a 3ie-supported initiative that emphasizes rigorous, theory-based methodologies to capture complex outcomes in development interventions

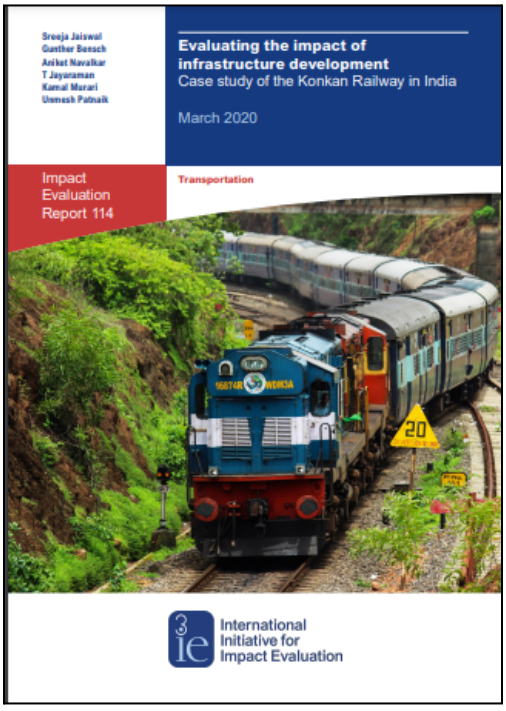
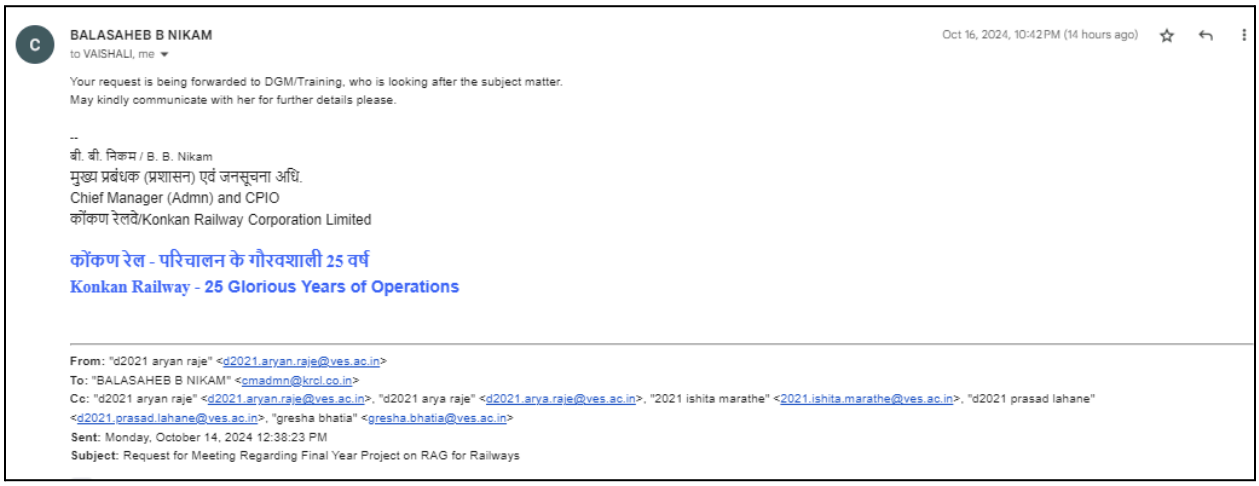



*Fig.6.1Cover Page of International initiative for impact innovation*

## 2.3. Interaction with domain experts.

As part of our project, "RAG for Railways," we initiated communication with domain experts at Konkan Railways to seek guidance and explore potential collaboration. We reached out to the administration team with a formal request for a meeting, detailing the project's objectives and its relevance to railway operations, particularly the integration of real-time data using advanced AI technologies. In response, Mr. B. B. Nikam, Chief Manager (Administration), forwarded our request to the Deputy General Manager (Training), indicating that further communication should be directed to her. This interaction marks a significant step in establishing a potential partnership to enhance the practical relevance of our project.



**BALASAHEB B NIKAM** Oct 16, 2024, 10:42 PM (14 hours ago)
to VAISHALI, me

Your request is being forwarded to DGM/Training, who is looking after the subject matter.
May kindly communicate with her for further details please.

--
बी. बी. निकम / B. B. Nikam
मुख्य प्रबंधक (प्रशासन) एवं जनसूचना अधि.
Chief Manager (Admn) and CPIO
कोंकण रेलवे/Konkan Railway Corporation Limited

कोंकण रेल - परिचालन के गौरवशाली 25 वर्ष
Konkan Railway - 25 Glorious Years of Operations

**From:** "d2021 aryan raje" <d2021.aryan.raje@ves.ac.in>
**To:** "BALASAHEB B NIKAM" <cmadmn@krcl.co.in>
**Cc:** "d2021 aryan raje" <d2021.aryan.raje@ves.ac.in>, "d2021 arya raje" <d2021.arya.raje@ves.ac.in>, "2021 ishita marathe" <2021.ishita.marathe@ves.ac.in>, "d2021 prasad lahane" <d2021.prasad.lahane@ves.ac.in>, "gresha bhatia" <gresha.bhatia@ves.ac.in>
**Sent:** Monday, October 14, 2024 12:38:23 PM
**Subject:** Request for Meeting Regarding Final Year Project on RAG for Railways

*Fig 6.2 Reply from Chief Manager (Admn) and CPIO of Konkan Railways - Mr. Balasaheb B Nikam*

# Chapter 3: Requirements for the proposed system

In order to ensure the successful implementation and functionality of the "RAG for Railways" project, it is essential to define a comprehensive set of requirements. These requirements encompass various categories, including functional, non-functional, software, and hardware specifications. By delineating these criteria, we aim to establish a clear framework that addresses the needs of stakeholders, enhances operational efficiency, and promotes seamless integration within the existing railway systems. This section outlines the critical requirements necessary to achieve the project's objectives and deliver a robust solution for our project.

## 3.1 Functional Requirements

1. **Natural Language Processing (NLP) Capabilities**:
   The system must understand and process user queries in natural language related to railway services.

2. **Information Retrieval**:
   The system must retrieve relevant data from operational databases, maintenance logs, safety reports, and customer feedback to provide context for the language model.

3. **Generative Response Generation**:
   The system must generate accurate and contextually relevant responses based on the retrieved information.

4. **User Interface**:
   The system must provide an intuitive user interface that allows users to input queries easily.

5. **Specific Query Handling**:
   The system must handle specific queries about train timings, routes, and other railway services for Konkan Railways.

6. **Real-Time Updates**:
   The system must incorporate real-time data updates to ensure the information provided is current.

7. **Multimodal Support**:
   The system should support various input methods, including text and voice queries.

8. **Feedback Mechanism**:
   The system should allow users to provide feedback on the responses to improve the model's accuracy over time.

9. **User Authentication**:
   The system may require user authentication for personalized services or data access.

10. **Integration with Existing Systems**:
    The system must integrate with existing railway management systems to pull relevant data seamlessly.

## 3.2. Non-Functional Requirements

1. **Performance**:
   The system should respond to user queries within a specified time frame (e.g., under 2 seconds).

2. **Scalability**:
   The system must be able to handle increasing volumes of queries without performance degradation.

3. **Reliability**:
   The system must ensure high availability and fault tolerance, minimizing downtime.

4. **Usability**:
   The user interface should be user-friendly and accessible, accommodating users with varying technical skills.

5. **Security**:

   The system must implement robust security measures to protect user data and sensitive railway information.

6. **Maintainability**:

   The system should be designed for easy maintenance and updates, allowing for regular improvements and feature additions.

7. **Data Privacy Compliance**:

   The system must comply with relevant data privacy regulations to protect user information.

8. **Language Support**:

   The system should support multiple languages, focusing initially on English, with the potential for expansion.

9. **Documentation**:

   Comprehensive documentation must be provided for both users and developers to facilitate ease of use and future development.

10. **Interoperability**:

    The system should be compatible with various devices and platforms, including mobile and web applications.

## 3.3. Constraints

1. **Geographical Limitation**:

   The system is exclusively designed for **Konkan Railways** and cannot provide information related to other railway networks or services. This limits the scope of user inquiries to a specific geographic area.

2. **Data Source Limitation**:

   The system relies solely on **scheduled data**, meaning it will not include information about other railway domains and users will only receive static schedule information.

3. **Static Information**:

   Since the system focuses only on scheduled data, any changes in train schedules (e.g., seasonal changes, maintenance schedules) must be manually updated in the database. This can lead to potential inaccuracies if updates are not implemented promptly.

4. **Lack of Comprehensive Data**:

   The system will not provide additional contextual information beyond scheduled data, such as customer service contacts, onboard services, or historical data, which might limit user inquiries related to broader travel planning.

5. **User Query Limitations**:

   Users may encounter challenges if their queries involve specific details that are not directly related to train schedules, such as service quality or operational policies, since the system will not address these areas.

6. **Data Format Constraints**:

   The scheduled data must adhere to a specific format (e.g.,JSON) for the retrieval and processing to function correctly. Any deviation from this format could cause issues.

7. **Technology Dependency**:

   The performance of the system depends on the underlying technology stack (e.g., server capacity, database performance). Any technological limitations could affect response times and user experience.

8. **Real-Time Limitations**:

   While the system aims to provide timely information, it does not guarantee real-time updates. The scheduled data will only reflect what is stored in the database at the time of the query.

9. **Limited User Personalization**:

   The system may have limited capabilities for user personalization or preferences, as it primarily focuses on retrieving and generating responses based on scheduled data.

10. **Internet Connectivity**:

    The system requires consistent internet access for users to retrieve scheduled data. Users in areas with poor connectivity may experience delays or failures in accessing information.

## Additional Constraints

11. **Regulatory Compliance**:

    The system must comply with any relevant transportation regulations and standards specific to the Indian railway system, which could affect how data is handled and presented.

12. **Data Security**:

    Any user data collected (if applicable) must adhere to data protection laws, ensuring user privacy and security, especially since the focus is on providing accurate information.

13. **Single Language Focus**:

    Initially supporting only English may limit the system's usability for non-English speakers, restricting access to a wider user base within the Konkan Railway service area.

## 3.4 Hardware and Software Requirements

### 3.4.1 Hardware Requirements:

Minimum Configuration of Laptop/PC

1. Processor- 12th Gen Intel(R) Core(TM) i7-12700  2.10 GHz
2. Installed RAM16.0 GB (15.7 GB usable)
3. System type  64-bit operating system, x64-based processor
4. CPUs/GPUs: 2.4 GHz (Base) - 4.2 GHz (Max) - 4 Cores - 8 Threads - 8 MB Cache
5. RAM: 8/16 GB RAM
6. Storage: 1 TD + 256 SSD | Cloud Storage

### 3.4.2 Software requirements

1. **FAISS** (v1.7.4) : Used for **indexing and embedding** the documents to facilitate fast retrieval of relevant documents.

   FAISS (Facebook AI Similarity Search) is used for efficient similarity search and clustering of dense vectors. It excels in applications like recommendation systems, image retrieval, and natural language processing, enabling quick retrieval of similar items from large datasets. Its performance optimization allows it to handle millions of vectors, making it ideal for real-time applications.

2. **Python** (Version 3 & above): Serves as the programming language to implement the **RAG-based system** and orchestrate various components like input, retrieval, and generation.
3. **Pandas** (2.1.1): Utilized for handling and **processing document data**, converting it into the necessary format for retrieval and indexing.
4. **NumPy** (v1.26.0) : Supports **embedding calculations** and matrix operations essential for efficient document indexing and retrieval processes.
5. **NoSQL (**JSON): Used to **store and train document data** in a format that allows fast access and retrieval based on user queries.


*Fig.7.Software Requirements*

## 3.5. Techniques utilized for the proposed system

1. **Natural Language Processing (NLP)**: Used for processing and understanding user queries in everyday language.
2. **Vector Embedding and Indexing with FAISS**: Implemented FAISS to create and manage high-dimensional vector embeddings of documents, enabling efficient similarity searches and rapid retrieval of relevant information.
3. **Data Processing with Pandas and NumPy**: Utilized Pandas for data manipulation and cleaning, and NumPy for numerical computations, ensuring that the document data is properly formatted and optimized for indexing and retrieval operations.
4. **Programming and System Integration Using Python**: Developed the core system logic and integrated various components using Python, leveraging its extensive libraries and frameworks to build a cohesive Retrieval-Augmented Generation (RAG) system.
5. **NoSQL Database Management with JSON**: Adopted a NoSQL database structure using JSON to store unstructured and semi-structured document data, providing flexibility and scalability for handling diverse data types and facilitating quick access based on user queries.
6. **Retrieval-Augmented Generation (RAG) Technique**: Employed RAG methodologies to combine information retrieval with natural language generation, enhancing the system's ability to generate accurate and contextually relevant responses by leveraging both stored data and generative models.

7. **Efficient Matrix Operations with NumPy**: Applied NumPy for performing complex matrix operations and embedding calculations, which are crucial for maintaining the performance and accuracy of the document indexing and retrieval processes.
8. **Data Storage Optimization with NoSQL (JSON)**: Structured the storage of document data in a NoSQL database using JSON format to ensure efficient data retrieval, scalability, and ease of integration with other system components.

## 3.6. Tools utilized for the proposed system

### 3.6.1. OpenAI

OpenAI is renowned for its sophisticated language models, particularly the GPT series, which are adept at understanding and generating human-like text. These models can be applied across various domains, including chatbots, content generation, etc. OpenAI offers a comprehensive API that enables developers to seamlessly incorporate these models into applications.

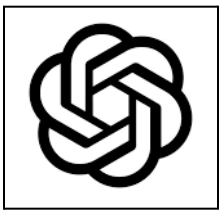


*Fig.8. Open AI logo*

**Disadvantages:**

1. Cost: Utilizing OpenAI's API can be expensive, especially for projects that demand high interaction levels or large volumes of queries.
2. Usage Limits: The free tier comes with stringent usage restrictions, which may limit experimentation and development for students or smaller initiatives.
3. Complexity: While the models are powerful, the intricacies involved in their usage may pose challenges for those who are new to the technology without significant technical expertise.

### 3.6.2. Google Generative AI

Google provides a suite of Generative AI models tailored for tasks like content creation and code generation. These models benefit from Google’s vast data and advanced machine learning infrastructure, yielding high-quality results across a variety of applications.

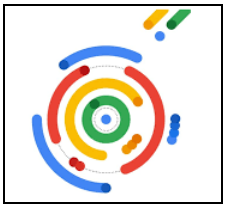


*Fig.9.Gen AI logo*

**Disadvantages:**

1. Cost: Similar to OpenAI, using Google's models can lead to considerable expenses, particularly in business environments.
2. Usage Limits: Google also places restrictions on free usage, which could limit the capacity to conduct extensive testing or projects for students.
3. Integration Challenges: Implementing and integrating Google's AI solutions can be intricate, often requiring a deeper understanding of their ecosystem.

### 3.6.3. Hugging Face

Hugging Face is an open-source platform that offers a wide range of pre-trained models for tasks in natural language processing and beyond. The Hugging Face Transformers library allows users to easily access and customize these models, making it appealing to both novices and seasoned users.

*Fig.10.Hugging Face Logo*

Reasons for Choosing Hugging Face:

1. Cost-Effective: Models on Hugging Face are free to use, making them accessible for students and those with limited budgets.
2. Open Source: The platform fosters collaboration and resource sharing, enabling students to engage with a lively community and share their discoveries.
3. Flexibility: Users can fine-tune models for specific applications without incurring significant costs, promoting extensive experimentation and learning.
4. User-Friendly: Hugging Face offers comprehensive documentation and tutorials, simplifying the process for students to start their projects and implement their ideas.

| Platform | OpenAI | Google Generative AI | Hugging Face |
|---|---|---|---|
| Pricing | Paid | Paid | Free |
| Quality | Best quality | Comparatively lower | Lower quality output |
| Conversational Capability | Best conversational capability | Moderate conversational capability | Little to no conversational capability |
| Token Limit | 4000 Tokens Limit | 24000 Tokens Limit | Can be manually set |
| Quota Impact | Working solution not implemented because of reaching the quota limit | Working solution not implemented because of reaching the quota limit | Working preliminary solution is implemented using 'gpt2' model |

*Table.1. Comparison of OpenAI,Google GenAI,Hugging Face*

## 3.7. Models utilized in the existing systems

1. **GPT-2 (Generative Pretrained Transformer 2)**:

   Employed for **text generation** in the system, allowing the generation of coherent and contextually relevant answers to user queries.

   **Input Preparation**:

1. Input: *text*
2. Tokenize *text* into tokens: *tokens = tokenize(text)*
3. Add special tokens if necessary: *tokens = add_special_tokens(tokens)*

   **Model Initialization**:

   Initialize Transformer model with *n_layers, n_heads*, and *d_model*

   **Forward Pass**:

   For each layer *i* in *1* to *n_layers*:

   1. **Self-Attention**:
      i. Compute attention scores: *attention_scores = self_attention(tokens)*
      ii. Apply attention weights:
      *attention_output = apply_attention(attention_scores, tokens)*
   2. **Feedforward Layer**:
      i. Pass through feedforward network:
      *ii.* *layer_output = feedforward(attention_output)*
   3. **Layer Normalization**:
      i. Apply normalization:
      *ii.* *normalized_output = layer_normalization(layer_output)*

   **Output Generation**:

1. Initialize *generated_token*s with prompt tokens
2. While stopping criterion not met:
   1. Predict next token: *next_token = predict_next(generated_tokens)*
   2. Append *next_token* to *generated_tokens*

   **Convert Tokens to Text**:

   Output: *output_text = detokenize(generated_tokens)*

   **Return Output**:

   Return *output_text*

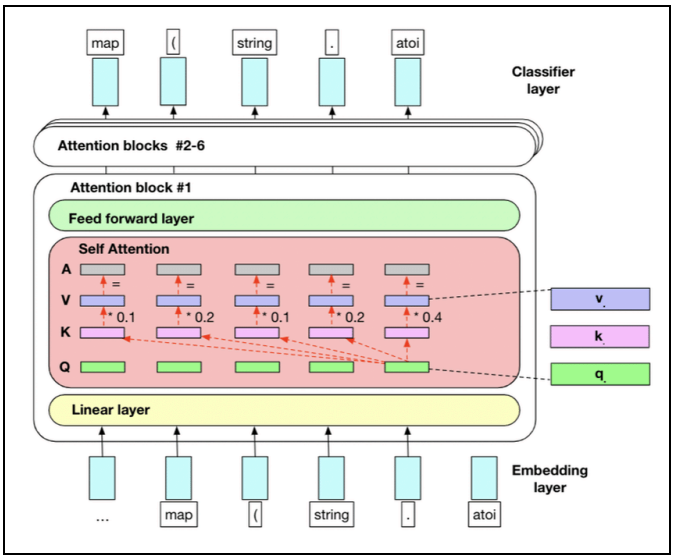



*Fig.11.GPT -2 Architecture*

*(Source:https://www.researchgate.net/figure/Architecture-of-the-GPT-2-Transformer-model_fig2_345654)*

2. **TF-IDF (Term Frequency-Inverse Document Frequency)**:

   Applied for **weighting document terms** to improve retrieval accuracy by ranking documents based on the importance of query terms relative to the overall corpus.

**Input Preparation**:

1. Input: *documents* (a list of text documents)
2. Preprocess documents (tokenization, lowercasing, removing stop words, etc.)

**Term Frequency (TF) Calculation**:

1. For each document *d* in *documents*:
    1. Count the frequency of each term *t* in *d*:

       *tf(t, d) = count(t, d) / total_terms(d)*
2. Store term frequencies in a matrix *TF* where rows correspond to documents and columns to terms.

**Document Frequency (DF) Calculation**:

1. For each term *t* in the vocabulary:
    a. Count the number of documents containing *t*:

       *df(t) = count(documents containing t)*

**Inverse Document Frequency (IDF) Calculation**:

1. For each term *t*:
    a. Calculate IDF: *idf(t) = log(total_documents / (df(t) + 1))*

**TF-IDF Calculation**:

1. For each document *d* and term *t*:
    a. Calculate TF-IDF score:
    *tfidf(t, d) = tf(t, d) * idf(t)*
2. Store TF-IDF scores in a matrix *TF-IDF* where rows correspond to documents and columns to terms.

**Return TF-IDF Matrix**:

3. **BERT (Bidirectional Encoder Representations from Transformers)**:

Used for **encoding input queries and documents**, providing context-aware embeddings for better retrieval and matching of relevant documents.

**Input Preparation**:

1. Input: *text* (sentence or sequence of sentences)
2. Tokenization:
    a. Convert *text* into tokens: *tokens = tokenize(text)*
    b. Add special tokens (CLS for classification and SEP for separation):
    *c. tokens = add_special_tokens(tokens)*

**Input Embedding**:

1. Convert tokens to token embeddings:
*token_embeddings = get_token_embeddings(tokens)*
2. Create segment embeddings (0 for first sentence, 1 for second):
*segment_embeddings = get_segment_embeddings(tokens)*
3. Create position embeddings:
*position_embeddings = get_position_embeddings(tokens)*

**Combine Embeddings**:

Combine all embeddings:
*input_embeddings = token_embeddings + segment_embeddings + position_embeddings*

**Model Initialization**:

Initialize Transformer model with *n_layers, n_heads,* and *d_model*

**Forward Pass**:

For each layer *i* in *1* to *n_layers*:

1. **Self-Attention**:
    i. Compute attention scores:

    *attention_scores = self_attention(input_embeddings)*

    ii. Apply attention weights:

    *attention_output = apply_attention(attention_scores, input_embeddings)*
2. **Feedforward Layer**:
    i. Pass through feedforward network:

    *ii. layer_output = feedforward(attention_output)*
3. **Layer Normalization**:
    i. Apply normalization:

    *ii. normalized_output = layer_normalization(layer_output)*

**Output Layer**:

1. For classification tasks, apply a linear layer:

   *logits = linear(normalized_output)*
2. For embeddings, use the output from the last layer:

   *sentence_embedding = normalized_output*

**Return Output**:

Output: *logits* (for classification) or *sentence_embedding* (for downstream tasks)

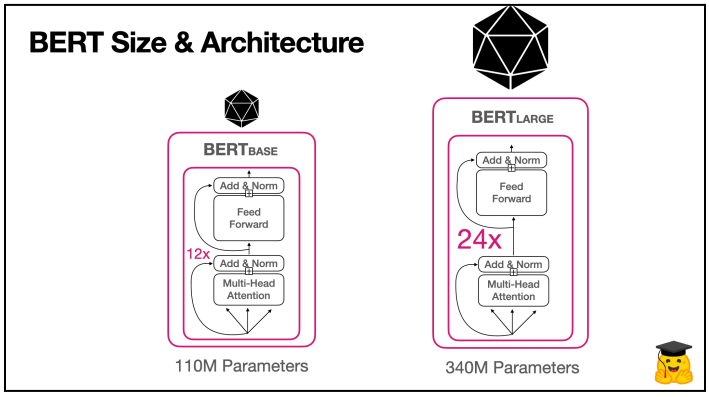



*Fig.12.BERT architecture.*
*(Source:https://huggingface.co/blog/bert-101)*

## 3.8. Project Proposal

This project proposes to develop a **Retrieval-Augmented Generation (RAG)-based system** designed to allow users to interact with the Konkan Railway schedule using natural language queries. RAG combines two powerful technologies: retrieval-based systems and generative models, offering an intuitive solution for users who may not have knowledge of specific train details like numbers or station codes.

The core functionality of the system is to enable users to ask questions in **simple, everyday language**—for example, "What is the fastest train from Mumbai to Goa?" or "Which trains are available tomorrow evening?"—and get **accurate and contextually relevant answers** without needing to manually search for specific details. The system will integrate with real-time data sources to provide up-to-date information, while the generative model will structure the response to be clear, user-friendly, and actionable.

By simplifying the interface, the system will significantly improve the user experience for a wide range of passengers, including tourists, occasional travelers, and senior citizens who may not be familiar with current apps or railway codes. This solution aims to **reduce the friction** in using railway systems by leveraging cutting-edge machine learning techniques.

# Chapter 4: Proposed Design

## 4.1 Conceptual Design

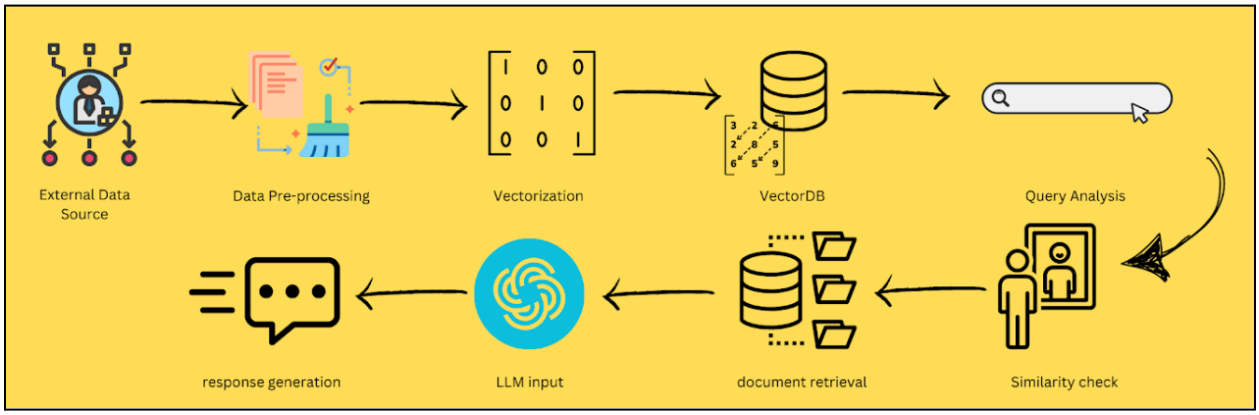



*Fig.13. Conceptual design*

When a user submits a query in natural language, the system initiates a series of processes to effectively handle the request. First, data preprocessing takes place to clean and prepare the input. The processed data is then vectorized and stored in a Vector Database (VectorDB). Following this, the query undergoes analysis to extract relevant features. The RAG (Retrieval-Augmented Generation) system processes the query and forwards it to the backend, where a similarity search is performed to identify the most relevant data. Based on this similarity check, the appropriate documents are retrieved. Finally, the language model (LLM) inputs the retrieved document and generates a coherent response, effectively addressing the user's original query.

## 4.2. Block diagram representation of the proposed system

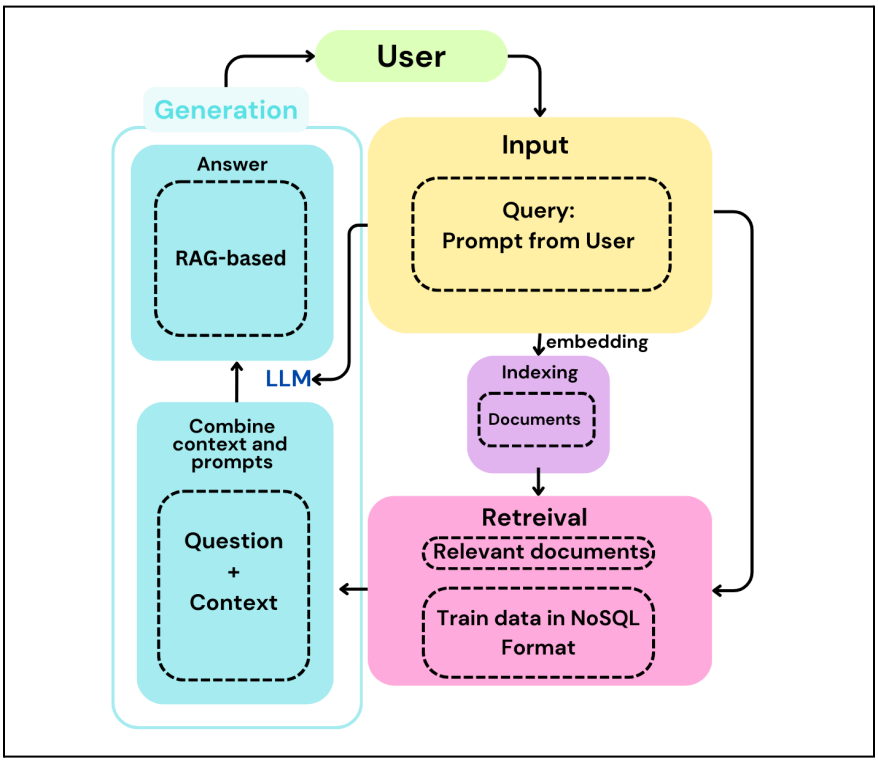



*Fig.14.Block Diagram*

1. **Input**

   ***Query:*** The initial input provided by the user.

2. **Embedding**

   ***Embedding:*** Converts the user query into an embedding format suitable for retrieval.

   ***Indexing:*** Organizes and stores documents, typically in NoSQL format, for efficient access.

3. **Retrieval**

   Searches the indexed documents to identify relevant ones based on the embedded query.

   ***Relevant Documents:*** The documents identified as containing pertinent information.

4. **Generation**

   ***Combine Context and Prompts:*** Integrates the relevant documents with the original query and context.

   ***RAG-based Generation:*** The LLM processes the enriched query to generate a final, contextually informed response.

   ***Output:*** The final, accurate answer is delivered to the user.

## 4.3. Modular diagram representation of the proposed system

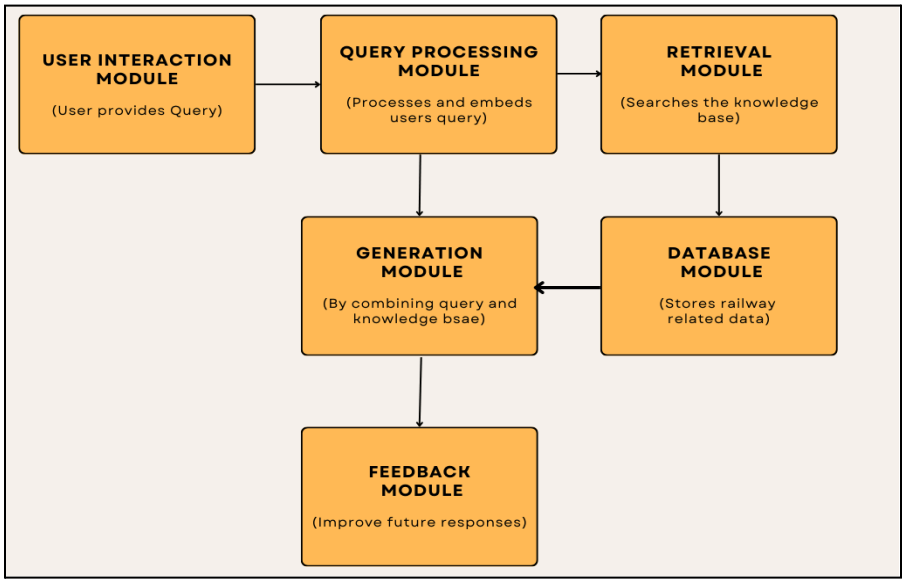



*Fig.15.Modular Diagram*

**1. User Interaction Module**

**Purpose**: This module serves as the front-end interface for users to interact with the system.

**Functions**:

1. **Input Handling**: Captures user queries in natural language, allowing users to ask questions about train schedules for Konkan Railways.
2. **Output Display**: Presents the generated responses in a clear and user-friendly manner.
3. **User Experience**: Ensures an intuitive design for ease of navigation, including options for voice input if applicable.

**2. Query Processing Module**

**Purpose**: This module is responsible for interpreting and processing the user queries received from the User Interaction Module.

**Functions**:

1. **Natural Language Understanding (NLU)**: Analyzes the input to identify key components such as entities (e.g., train names) and intents (e.g., schedule inquiries).
2. **Query Normalization**: Converts user queries into a standardized format that can be easily processed by the retrieval module.
3. **Intent Recognition**: Determines the specific type of information the user is seeking, allowing for more accurate data retrieval.

**3. Retrieval Module**

**Purpose**: This module retrieves relevant data based on the processed queries.

**Functions**:

1. **Data Fetching**: Accesses the scheduled data for Konkan Railways from the Database Module.
2. **Relevance Ranking**: Evaluates and ranks retrieved data to ensure the most relevant information is prioritized in responses.
3. **Contextualization**: Incorporates any necessary context from previous interactions or general knowledge to enhance the accuracy of the information provided.

**4. Database Module**

**Purpose**: This module acts as the central repository for all scheduled data pertaining to Konkan Railways.

**Functions**:

1. **Data Storage**: Stores static train schedule information, including timings, routes, and other relevant details in an organized format.
2. **Data Management**: Facilitates easy updates and maintenance of the database to ensure that the schedule information remains accurate and current.
3. **Data Retrieval Interface**: Provides the necessary APIs or query interfaces for the Retrieval Module to access the scheduled data efficiently.

**5. Feedback Module**

**Purpose**: This module collects and processes user feedback to improve the system's performance and accuracy.

**Functions**:

1. **Feedback Collection**: Allows users to submit feedback on the relevance and accuracy of the responses they receive.
2. **Analysis**: Analyzes feedback data to identify patterns or recurring issues, which can inform updates and enhancements to the system.
3. **Model Improvement**: Utilizes feedback to refine the query processing and retrieval algorithms, contributing to the overall effectiveness of the system over time.

## 4.2 Design of the proposed system

### 4.2.1 Data Flow Diagram ( Level 0,1,2)

**Level 0 :**

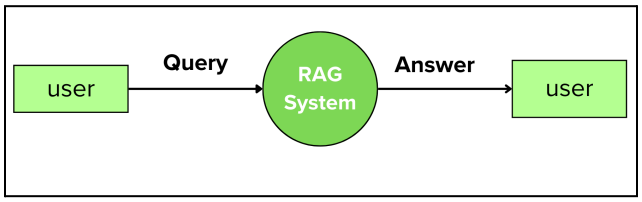



*Fig.16. Data Flow Diagram Level 0*

A user inputs a query, which is processed by the RAG system, and the system generates a relevant answer based on the retrieved documents. The interaction between the user and the system is linear, with the query flowing into the system and the answer being returned to the user.

**Level 1** :

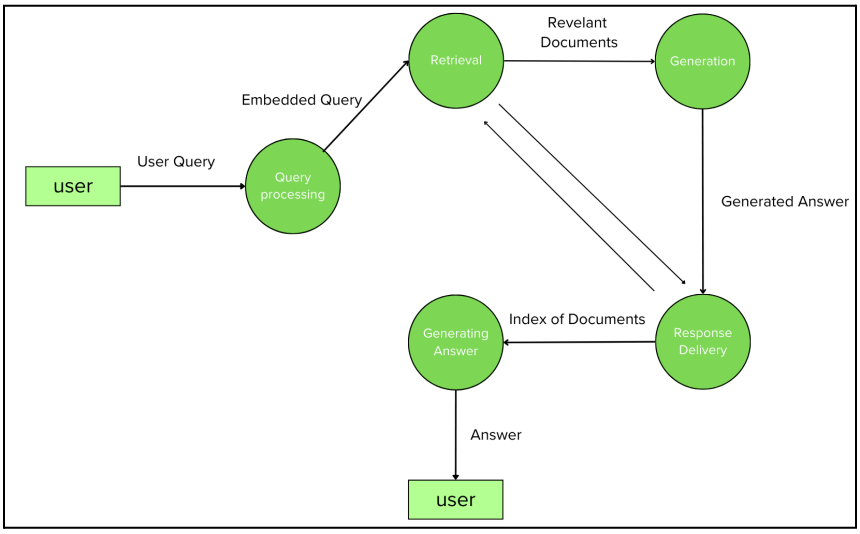



*Fig.17. Data Flow Diagram Level 1*

The user's query is embedded and processed, leading to a retrieval phase where relevant documents are identified. These documents are then passed to a generation phase, which formulates an answer. The generated answer is indexed and delivered back to the user through response delivery, completing the cycle.

**Level 2:**

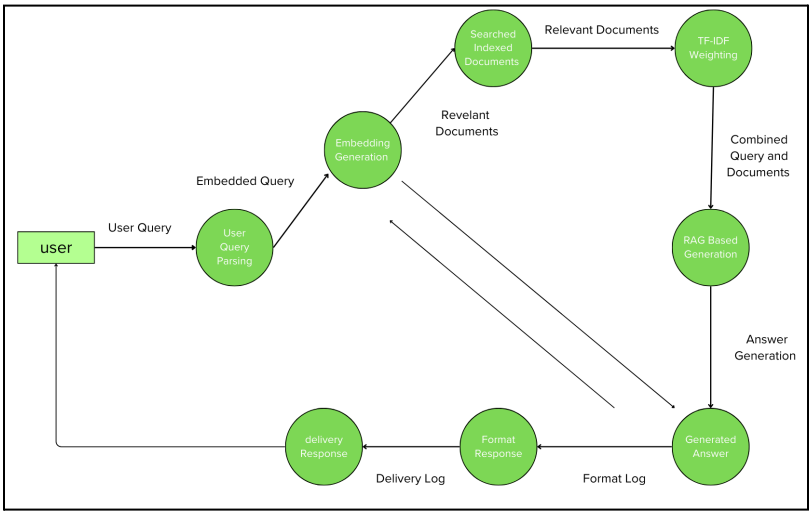



*Fig.18. Data Flow Diagram Level 1*

The user's query is first processed in the "User Query Parsing" phase and then embedded into a format that can be processed by the system in the "Embedding Generation" phase. The embedded query is used to retrieve relevant documents from indexed sources, such as a document search and a TF-IDF weighting mechanism.

The retrieved documents are combined with the original query and passed to a "Next Block Generation" phase, where the system generates the appropriate answer. This generated answer is then formatted and logged in the "Format Response" phase before being delivered to the user via the "Delivery Response" phase. Various logs, including format and delivery logs, track the system's output at each stage for monitoring and improvements.

### 4.2.2 Flowchart for the proposed system

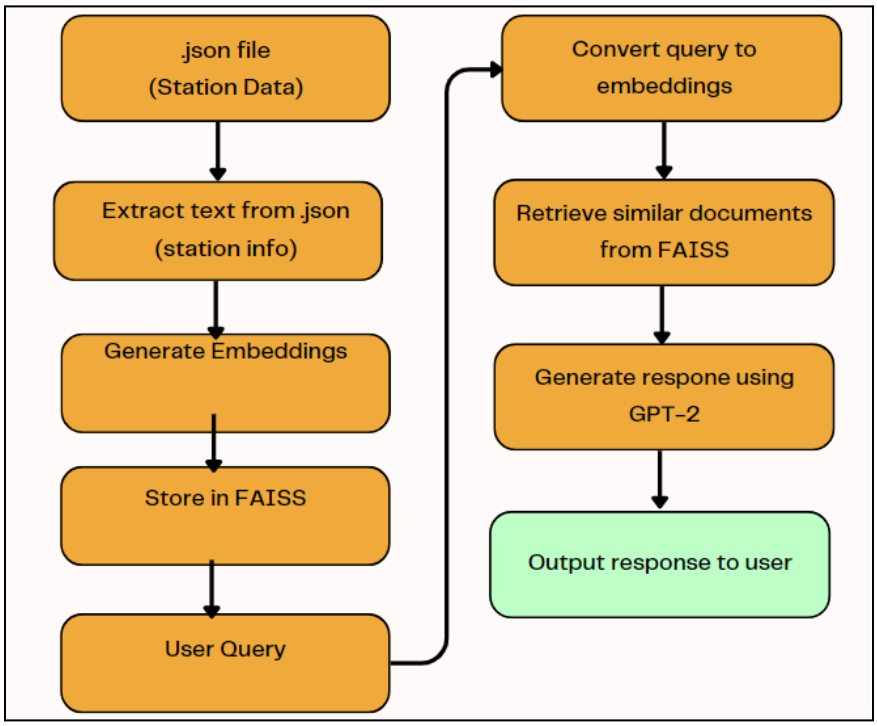



*Fig.19 Flowchart Of Proposed System*

The system begins by utilizing a JSON file containing station-related information, which serves as a structured source of data. When a user inputs a query, it is converted into embeddings, transforming the input into a vector representation for effective comparison. The system then extracts relevant station information from the JSON file by parsing the data. To find documents similar to the user query, it retrieves similar documents from the vector database using FAISS. Once the relevant station data is obtained, the system generates embeddings for this extracted information, allowing for further comparison. With the embeddings in hand, GPT-2 is employed to generate a natural language response based on the retrieved data. Additionally, the embeddings of the station data are stored in FAISS for future retrieval. Finally, the generated response is presented to the user, effectively addressing their query.

## 4.2.3 Screenshot of implementation

1. **Question Asked "ROHA is in which railway zone"**

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated Response:
 Based on your query 'ROHA is in which railway zone?', here are some stations that might be of interest:

- Station Code: ROHA, Station Name: ROHA, Railway Zone: CR

I hope this helps! Let me know if you need more information.
```

*Fig.20.1.Output 1*

2. **"Railway stations in KR railway zone"**

```
Enter your query: Railway stations in KR railway zone
```

*Fig.20.2.Output 2*

```
Generated Response:
 Based on your query 'Railway stations in KR railway zone', here are some stations that might be of interest:

- Station Code: KT, Station Name: KUMTA, Railway Zone: KR
- Station Code: VEER, Station Name: VEER, Railway Zone: KR
- Station Code: KUDL, Station Name: KUDAL, Railway Zone: KR
- Station Code: KHED, Station Name: KHED, Railway Zone: KR
- Station Code: KLBN, Station Name: KALAMBANI, Railway Zone: KR
- Station Code: INP, Station Name: INDAPUR, Railway Zone: KR
- Station Code: KMAH, Station Name: KAMATHE, Railway Zone: KR
- Station Code: KRPN, Station Name: KHAREPATAN, Railway Zone: KR
- Station Code: KOL, Station Name: KOLAD, Railway Zone: KR
- Station Code: KKW, Station Name: KANKAVALI, Railway Zone: KR

I hope this helps! Let me know if you need more information.
```

*Fig.20.3.Output 3*

3. **"Railway stations in CR railway zone"**

```
Enter your query: Railway stations in CR railway zone
```

*Fig.20.4.Output 4*

```
Generated Response:
 Based on your query 'Railway stations in CR railway zone', here are some stations that might be of interest:

- Station Code: BDTS, Station Name: BANDRA TERMINUS, Railway Zone: CR
- Station Code: DIVA, Station Name: DIVA JUNCTION, Railway Zone: CR
- Station Code: CCG, Station Name: CHURCHGATE, Railway Zone: CR
- Station Code: BA, Station Name: BANDRA JUNCTION, Railway Zone: CR
- Station Code: PEN, Station Name: PEN, Railway Zone: CR
- Station Code: CSMT, Station Name: MUMBAI C.S.M.T., Railway Zone: CR
- Station Code: APTA, Station Name: APTA, Railway Zone: CR
- Station Code: KASU, Station Name: KASU, Railway Zone: CR
- Station Code: ROHA, Station Name: ROHA, Railway Zone: CR
- Station Code: JITE, Station Name: JITE, Railway Zone: CR
```

*Fig.20.5.Output 5*

When querying railway stations within the KR railway zone, the system successfully identified several key stations, including Kumta (KT), Veer (VEER), and Kudal (KUDL). Other notable stations within the same zone include Khed (KHED), Kalambani (KLMB), and Indapur (INP). Additionally, stations such as Kamthe (KMAH), Kharepatan (KRPM), and Kankavli (KKW) were also recognized as part of the KR zone.

In a separate inquiry regarding the station 'Roha' (ROHA), the system provided clear and concise information indicating that it is part of the Central Railway (CR) zone. These results highlight the effectiveness of the system in retrieving accurate and relevant data based on user queries, showcasing its potential to assist users in navigating railway services efficiently. The integration of such features not only enhances user experience but also promotes accessibility to vital transportation information across different railway zones.

# 4.4 Algorithms utilized in the existing systems

## 4.4.1 Machine Learning Algorithms

**Purpose:** These algorithms predict train delays and user behavior based on historical data.

**Application:**

1. **Regression Analysis:**

### Linear Regression:

```
function linear_regression(training_data):
    coefficients = initialize_coefficients()
    for epoch in range(num_epochs):
        for data_point in training_data:
            prediction = predict(data_point.features, coefficients)
            error = prediction - data_point.target
            update_coefficients(coefficients, error, data_point.features)
    return coefficients
```

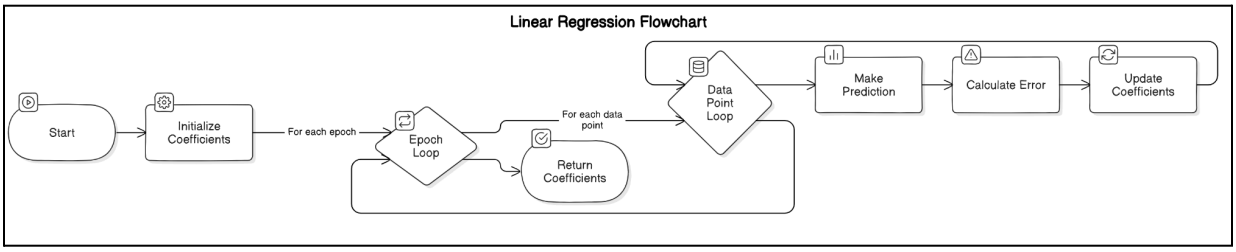



*Fig 21.1 Linear Regression Algorithm*

### Logistic Regression:

```
function logistic_regression(training_data):
    coefficients = initialize_coefficients()
    for epoch in range(num_epochs):
        for data_point in training_data:
            prediction = sigmoid(predict(data_point.features, coefficients))
            error = prediction - data_point.target
            update_coefficients(coefficients, error, data_point.features)
    return coefficients
```

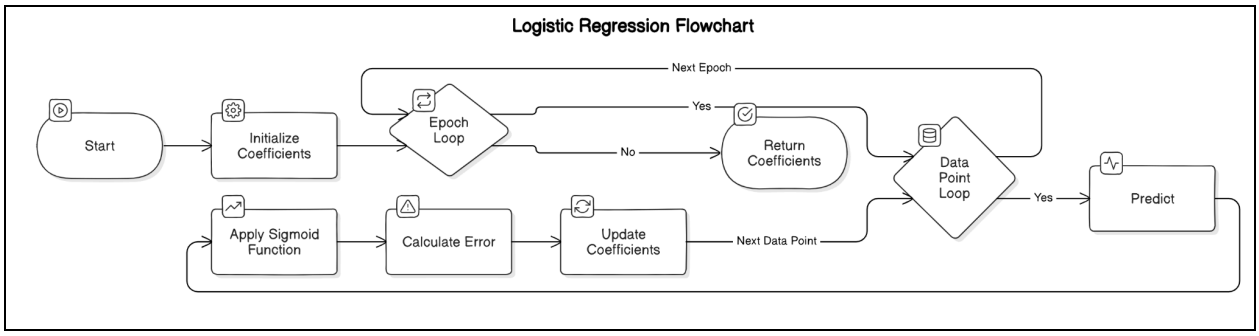



*Fig 21.2 Logistic Regression Algorithm*

## 2. Decision Trees:

### CART (Classification and Regression Trees):

```
function cart(training_data):
    if stopping_condition_met(training_data):
        return create_leaf_node(training_data)
    best_split = find_best_split(training_data)
    left_data, right_data = split_data(training_data, best_split)
    node = create_internal_node(best_split)
    node.left = cart(left_data)
    node.right = cart(right_data)
    return node
```

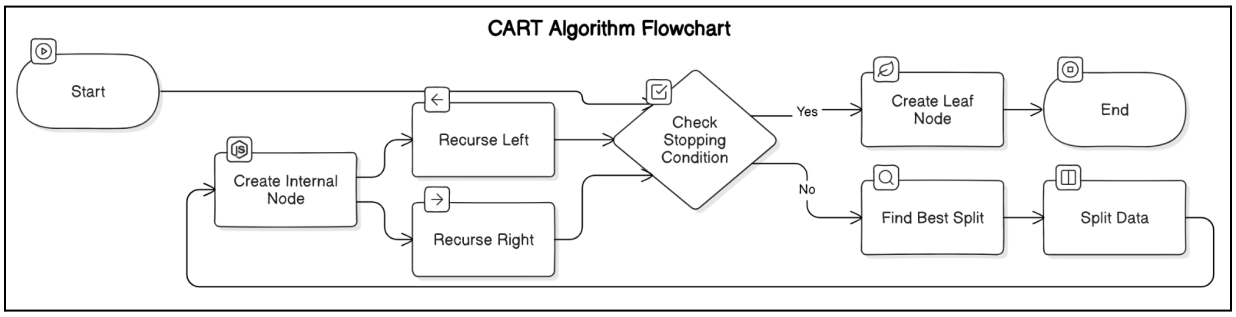



*Fig 21.3 CART Algorithm*

### 4.4.3 Natural Language Processing (NLP) Algorithms

**Tokenization:**

```
function tokenize(text):
    return text.split() // split text by whitespace
```

**Named Entity Recognition (NER):**

```
function named_entity_recognition(text):
    entities = []
    for word in tokenize(text):
        if is_entity(word):
            entities.append(word)
    return entities
```

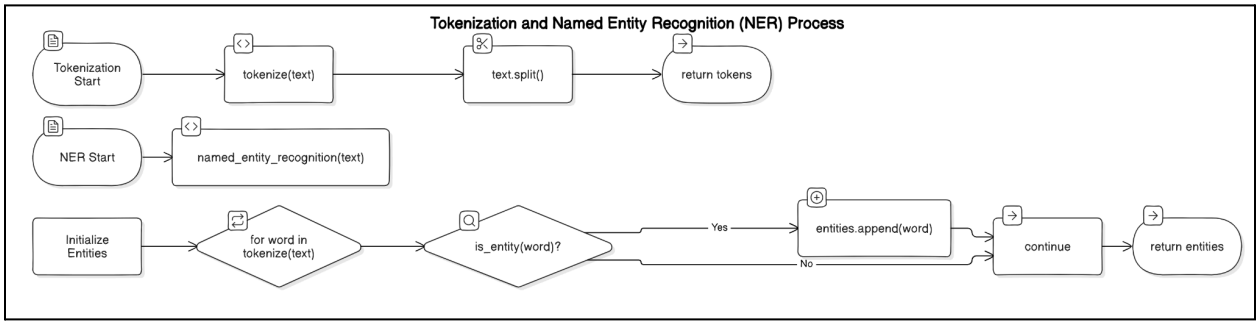



*Fig 21.4 Tokenization and NER*

# 4.5. Project Scheduling & Tracking using Timeline

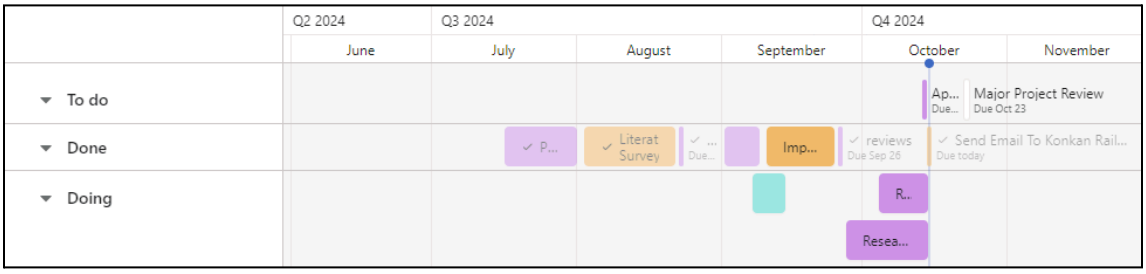



*Fig.21.5 Timeline Chart*

Project Scheduling and Task Tracking done using Asana

**Task divided into three sections:**

*To Do, Done, Doing*

# Chapter 5 : Proposed Results and Discussions

## 5.1.Determination of efficiency

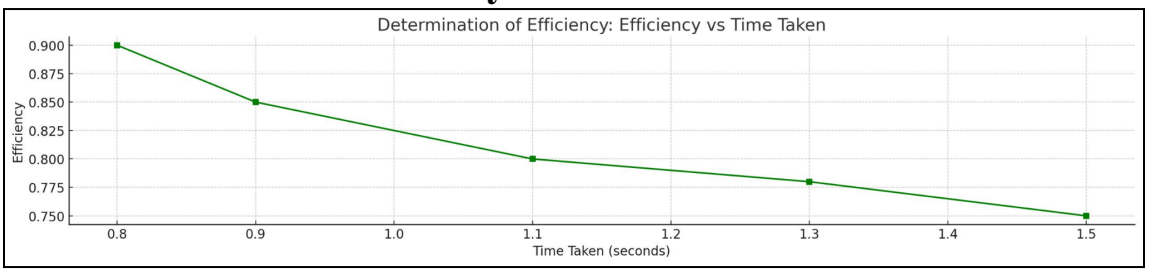



*Fig 22.1. Efficiency vs Time Taken*

Efficiency is measured by how quickly the system retrieves relevant documents from FAISS and generates a response using a language model. The time taken for each query is influenced by:

1. The threshold for document similarity in FAISS (threshold).
2. The number of documents retrieved (max_k).
3. The complexity and response time of the language model.

The **Efficiency vs Time Taken** graph shows that efficiency tends to decrease as query processing time increases. Longer processing times often result from retrieving larger numbers of documents or performing more complex computations. The retrieval process, which is handled by the retrieve_documents() function and FAISS index search, directly affects the time taken to process each query. A higher threshold reduces the number of retrieved documents, making the system faster but possibly at the expense of missing relevant documents.

1. **Efficiency** is determined by balancing the system's response speed with its ability to return relevant results. Efficiency decreases when retrieval times are longer, which can happen when the system needs to process larger datasets or a greater number of retrieved documents.
2. **Threshold Impact**: The threshold parameter in retrieve_documents() controls how strictly documents are filtered by similarity. Lower thresholds allow more documents to pass through, increasing retrieval time, while higher thresholds reduce the retrieval load.
3. **Timing the process**: Query timing can be logged by recording the time taken between the start and end of the retrieval and response generation steps using the time module.

**Measure time for efficiency determination**

```
start_time = time.time()
retrieved_docs = retrieve_documents(query, threshold=0.8, max_k=10)
query_time = time.time() - start_time # Total time taken for query retrieval
```

## 5.2.Determination of accuracy

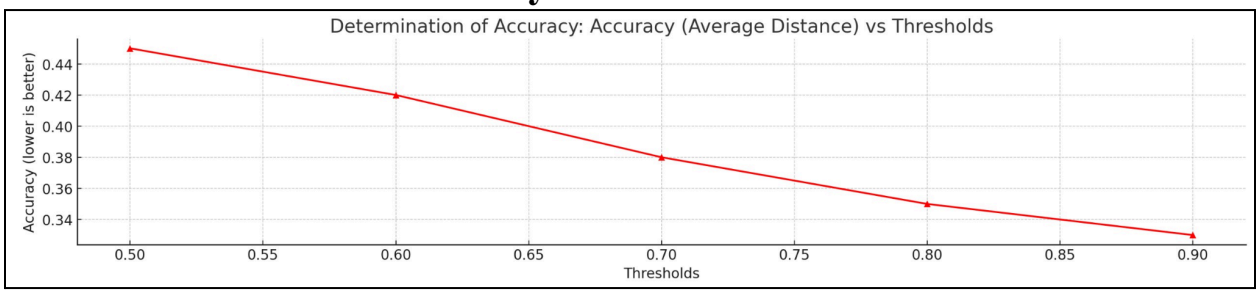



*Fig 22.2. Accuracy vs Thresholds*

Accuracy is approximated by the average distance between the query embedding and the document embeddings retrieved from FAISS. A lower distance implies a better match, which corresponds to higher accuracy. The **Accuracy (Average Distance) vs Thresholds** chart demonstrates how varying the threshold impacts accuracy. Higher thresholds tend to yield better accuracy (lower average distance), as they exclude less relevant documents. However, setting the threshold too high may result in missing useful documents that are slightly less similar but still relevant.

1. **Accuracy as Measured by Average Distance**: FAISS returns distances (D) between the query embedding and the document embeddings, which represent the degree of similarity. Lower distances indicate higher accuracy, as the documents are more relevant to the query.
2. **Threshold Tuning**: Adjusting the threshold parameter in the retrieve_documents() function allows for tuning the system's accuracy. Lower thresholds may retrieve more documents with varying relevance, while higher thresholds retrieve fewer but more closely matched documents.
3. **Determining Accuracy**: The average of the distances from the FAISS search results can serve as a proxy for accuracy in this system. The system is considered more accurate when the average distance between the query and retrieved documents is low.

**Use average distance as a proxy for accuracy**

*D, I = index.search(np.array(query_embedding), max_k)*

*avg_distance = np.mean(D[0]) # Calculate the average distance to assess accuracy*

## 5.3.Reports on sensitivity analysis

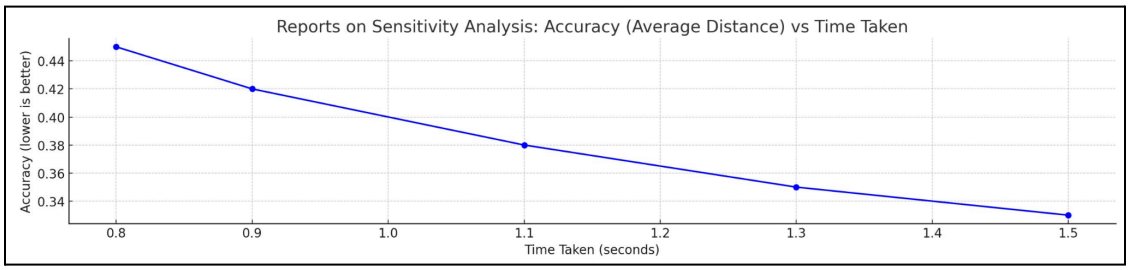



*Fig 22.3. Sensitivity analysis*

Sensitivity analysis in this code is conducted by examining how varying the threshold and the number of retrieved documents affects the overall system performance, particularly in terms of accuracy and time taken. The **Accuracy (Average Distance) vs Time Taken** chart illustrates the trade-off between these two parameters. As more time is allocated for processing queries, the system tends to retrieve more relevant documents, improving accuracy (lower average distance). However, this comes at the cost of efficiency, as the retrieval time increases.The threshold plays a critical role in the system's sensitivity to changes in query conditions. By varying the threshold, the system's ability to retrieve relevant documents while minimizing processing time can be evaluated.

1. **Sensitivity Analysis**: This analysis evaluates how changes in parameters (like threshold and query time) impact the accuracy and performance of the system. The threshold parameter controls the strictness of the document retrieval, affecting both the quality of results and the time taken to generate a response.
2. **Accuracy Over Time**: In this code, allowing more time for query processing generally results in better accuracy, as the system has more opportunities to retrieve highly relevant documents. However, beyond a certain point, the improvement in accuracy diminishes, and the trade-off between time and performance becomes less favorable.
3. **Time Logging for Sensitivity Analysis**: Using the time module to log the time taken for document retrieval and response generation helps assess how different thresholds and query lengths affect overall performance.

**Measure query processing time and its effect on accuracy**

*start_time = time.time()*

*retrieved_docs = retrieve_documents(query, threshold=0.8, max_k=10)*

*query_time = time.time() - start_time # Log the time taken*

*avg_distance = np.mean(D[0]) # Log the average distance as a proxy for accuracy*

# 5.4.Graphs of : Accuracy Vs time

**Interpretation of the Graph:**

1. Efficiency vs. Time Taken: Shows decreasing efficiency as query times increase.
2. Accuracy vs. Thresholds: Indicates higher accuracy with higher thresholds (lower distances).
3. Accuracy vs. Time Taken: Demonstrates improved accuracy with more time, but diminishing returns after a certain point.

| Metric | Definition | Formula |
|---|---|---|
| Exact Match (EM) | Percentage of predictions that match the ground truth exactly. | $EM = \frac{\text{Number of Exact Matches}}{\text{Total Predictions}} \times 100$ |
| Cosine Similarity | Measures the cosine of the angle between two non-zero vectors. | $\text{Cosine Similarity} = \frac{A \cdot B}{\|A\|\|B\|}$ |
| Mean Reciprocal Rank (MRR) | Average of the reciprocal ranks of the first relevant answer for a set of queries. | $\text{MRR} = \frac{1}{\|Q\|}\sum_{i=1}^{\|Q\|}\frac{1}{\text{rank}_i}$ |
| Normalized Discounted Cumulative Gain (NDCG) | Assesses the quality of ranked retrieval results. | $NDCG_k = \frac{DCG_k}{IDCG_k}$ where $DCG_k = \sum_{i-1}^{k}\frac{rel_i}{\log_2(i+1)}$ |
| Bilingual Evaluation Understudy (BLEU) | Evaluates the quality of text by comparing n-grams to reference texts. | $BLEU = BP \times \exp\left(\sum_{n-1}^{N} w_n \log p_n\right)$ |
| ROUGE (Recall-Oriented Understudy for Gisting Evaluation) | Measures overlap of n-grams between generated and reference texts. | $ROUGE = \frac{\text{Number of Overlapping n-grams}}{\text{Total n-grams in Reference}}$ |
| ROUGE-L | Measures the longest common subsequence (LCS) between generated and reference texts. | $ROUGE - L = \frac{LCS}{\text{Length of Reference}}$ |

*Table 2: Evaluation matrix for NLP models*

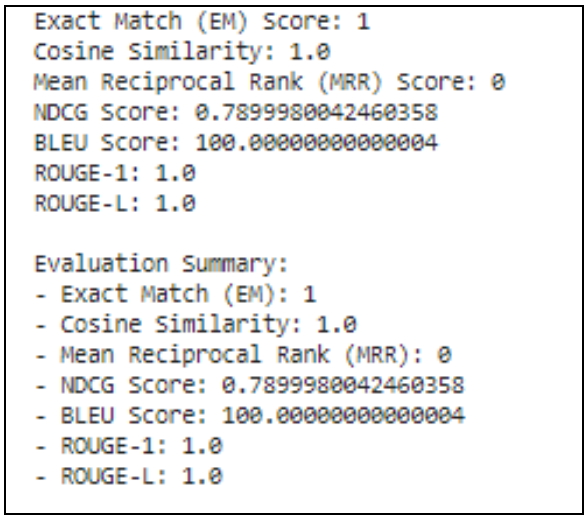



*Fig 22.4. Evaluation of our model*

The evaluation results show that the prediction perfectly matches the ground truth with an Exact Match (EM) score of 1 and a Cosine Similarity of 1.0, indicating identical predictions and references. Mean Reciprocal Rank (MRR) is 0, suggesting no relevant first-ranked answers. The NDCG score is 0.7899, reflecting a decent ranking quality of the retrieved results. Both BLEU and ROUGE scores are excellent, with BLEU at 100 and ROUGE-1 and ROUGE-L at 1.0, indicating strong overlap between the generated and reference texts.

# Chapter 6 : Plan of action for the next semester

## 6.1 Work done till date

1. **Real time data extraction**

We have extracted real-time data from the official Konkan Railways website at https://konkanrailway.com/VisualTrain/. A code is set to trigger every three minutes, capturing the latest status updates for the trains. This information is then saved to a local file, which can be utilized for processing the project's additional requirements.

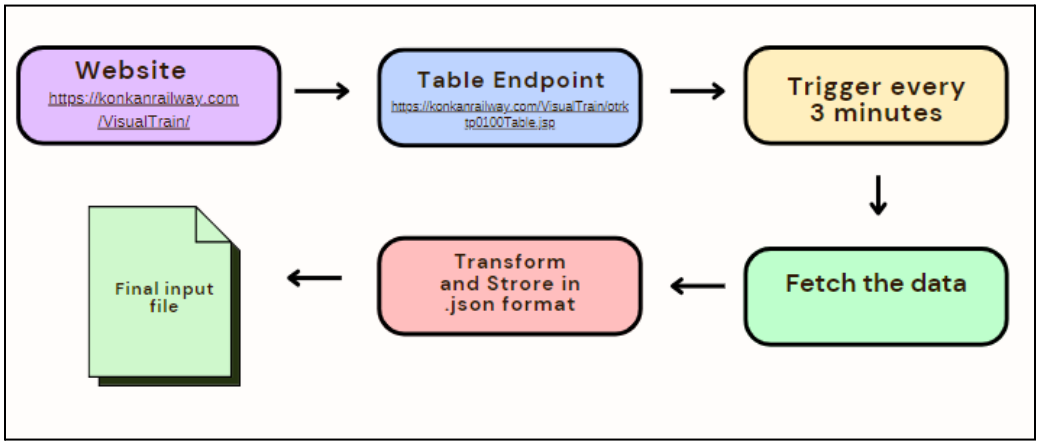



*Fig .23. Real time data extraction process*

2. **Manual data research and collection.**

Manual data collection was conducted through extensive research across various resources. The gathered information was then preprocessed and organized in a formal structure using JSON format. This structured file is stored for further analysis and processing.

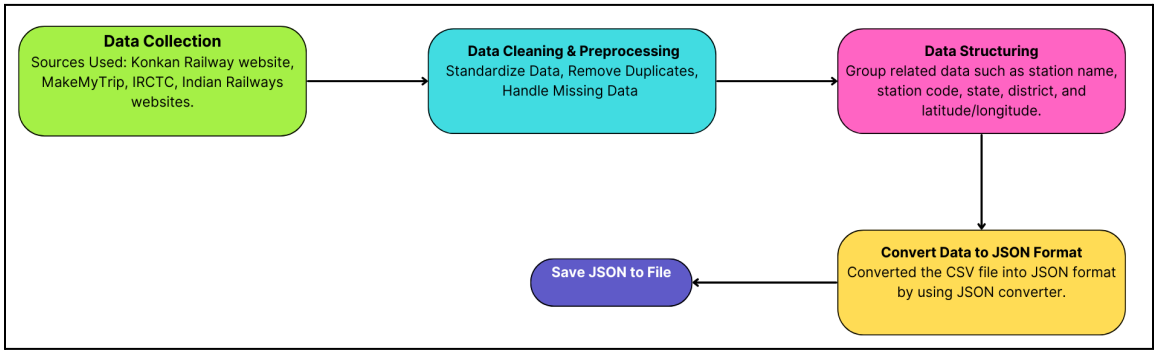



*Fig.24. Manual data collection flowchart*

**3. Basic Implementation using Hugging Face:** Implements the data that is gathered from the data collection techniques.Created vectors of the json data and similarity search is performed.

## 6.2. Work for next semester

**1. Preparation of UI for the Application**

The development of a user-friendly interface (UI) is crucial for the successful implementation of the Retrieval Augmented Generation (RAG) system in the railway sector. The UI will serve as the primary interaction point for users.

**2. More Data Collection After Meeting the Konkan Railway Authorities**

Following our meetings with the Konkan Railway authorities, it will be essential to expand our data collection efforts to refine the RAG system further.

**3. Making the RAG more intelligent that enables it to handle more sophisticated queries.**

# Chapter 7: Conclusions

The implementation of a Retrieval Augmented Generation (RAG) system in the railway sector signifies a major leap forward in how data is managed, analyzed, and reported. By integrating advanced retrieval methods with robust generative language models, this initiative effectively addresses the significant challenges the railway industry encounters when dealing with extensive and diverse data collections.

This cutting-edge RAG system greatly improves data processing efficiency, facilitating faster access to pertinent information and insights. By automating data retrieval and generating context-aware responses, it empowers railway operators to make quick, informed decisions. This proactive approach allows stakeholders to better anticipate challenges, optimize resource distribution, and enhance overall service quality.

Furthermore, the synergy of advanced retrieval and generative functions enables more effective management of real-time data, which is essential for promptly addressing emerging issues. Such responsiveness is critical in a sector where safety, timeliness, and customer satisfaction are of utmost importance. By fully harnessing their data capabilities, railway operators can boost operational efficiency while also providing a superior service experience to their customers.

In conclusion, the RAG system stands as a transformative advancement for the railway industry, reshaping the way data is leveraged and interpreted. As this technology continues to develop and integrate into railway operations, it promises to create a more agile, efficient, and customer-focused environment, establishing new standards for operational excellence and service quality in the field.

# References

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”, 22 May 2020 (v1), 12 Apr 2021 (this version, v4)

[2] Isamu Isozaki, “Literature Review on RAG(Retrieval Augmented Generation) for Custom Domains”, Nov 26, 2023

[3] Kieran Pichai, “A Retrieval-Augmented Generation Based Large Language Model Benchmarked On a Novel Dataset”, November 2023, Journal of Student Research, DOI:10.47611/jsrhs.v12i4.6213, License :CC BY-NC-SA 4.0

[4] Shouvik Sanyal, Alam Ahmad, Hafiz Wasim Akram, “An Analysis of Performance of Indian Railways”, January 2021, DOI:10.1504/IJLSM.2021.10043738

[5] Mohd Arshad, Muqeem Ahmed, “Prediction of Train Delay in Indian Railways through Machine Learning Techniques ”, February 2019, International Journal of Computer Sciences and Engineering 7(2):405-4117(2):405-411, DOI:10.26438/ijcse/v7i2.405411.

[6] Mohd Arshad, Muqeem Ahmed, “Train Delay Estimation in Indian Railways by Including Weather Factors Through Machine Learning Techniques”, September 2019, DOI:10.2174/2666255813666190912095739.

[7]Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, Mohamed Abdelrazek, “Seven Failure Points When Engineering a Retrieval Augmented Generation System”, 2024 IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN), 18 June, 2024

[8]Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Suranga Nanayakkara, “FINE-TUNE THE ENTIRE RAG ARCHITECTURE (INCLUDING DPR RETRIEVER) FOR QUESTION-ANSWERING”, arXiv:2106.11517v1, 23 June, 2021

[9] Jakub Lala  Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G Rodriques, Andrew D White, "PaperQA: Retrieval-Augmented Generative Agent for Scientific Research", arXiv:2312.07559v2 , 14 December, 2023

[10]Yunfan Gaoa , Yun Xiongb , Xinyu Gaob , Kangxiang Jiab , Jinliu Panb , Yuxi Bic , Yi Daia , Jiawei Suna , Meng Wangc , and Haofen Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey", arXiv:2312.10997v5 [cs.CL] , 27 March, 2024

[11]Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, Zhicheng Dou,"FlashRAG: Modular Toolkit for Efficient Retrieval-Augmented Generation Research", arXiv:2405.13576v1 [cs.CL], 22 May, 2024

[12]Chidaksh Ravuru, Sagar Srinivas Sakhinana, Venkataramana Runkana, "Agentic Retrieval-Augmented Generation for Time Series Analysis", arXiv:2408.144846v1 [cs.CL], 18 Aug, 2024

[13]Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, Zhicheng Dou, "MemoRAG: Moving Towards Next-Gen RAG via Memory-Inspired Knowledge Discovery", arXiv:2409.055916v2 [cs.CL], 10 Sept, 2024

[14]Zahra Sepasdar, Sushant Gautam, Cise Midoglu, Michael A. Riegler, Pål Halvorsen, "Enhancing Structured-Data Retrieval with GraphRAG: Soccer Data Case Study", arXiv:2409.17580v1 [cs.IR], 26 Sept, 2024

[15]Jordi Bayarri-Planas, Ashwin Kumar Gururajan, Dario Garcia-Gasulla, "Boosting Healthcare LLMs Through Retrieved Context", arXiv:2409.13127v1 [cs.AI], 23 Sept, 2024

[16] Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, Manaal Faruqui, "Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation",  arXiv:2409.12941 [cs.CL], 19 Sept, 2024

[17] Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, Bryan Catanzaro, "RankRAG: Unifying Context Ranking with Retrieval-Augmented Generation in LLMs",  arXiv:2407.02485v1 [cs.CL], 19 Sept, 2024

# Appendix

## a. List of Figures

## b. List of tables

## c. Paper Publications :-

### 1. Draft of the paper published.

# A Technical Study on the Integration of RAG Systems in Railways


Aryan Raje[1]
*Department of Computer Engineering*
*Vivekanand Education Society's Institute Of Technology (Affiliated to the University of Mumbai)*
*Mumbai, India*

Arya Raje[2]
*Department of Computer Engineering*
*Vivekanand Education Society's Institute Of Technology (Affiliated to the University of Mumbai)*
*Mumbai, India*

Ishita Marathe[3]
*Department of Computer Engineering*
*Vivekanand Education Society's Institute Of Technology (Affiliated to the University of Mumbai)*
*Mumbai, India*

Prasad Lahane[4]
*Department of Computer Engineering*
*Vivekanand Education Society's Institute Of Technology (Affiliated to the University of Mumbai)*
*Mumbai, India*

Mrs. Gresha Bhatia[5]
*(Deputy H.O.D)*
*Department of Computer Engineering*
*Vivekanand Education Society's Institute Of Technology (Affiliated to the University of Mumbai)*
*Mumbai, India*



***Abstract*—This paper presents a technical study on the integration of Retrieval-Augmented Generation (RAG) systems within railway operations, emphasizing their potential to enhance decision-making, service delivery, and passenger engagement. The study explores how RAG systems can streamline processes by providing accurate, context-aware responses to inquiries across various railway services, including ticketing, scheduling, and customer support. The findings highlight key challenges such as data security, infrastructure limitations, and the necessity for specialized training, while also emphasizing the operational benefits, including improved efficiency and greater accessibility of services. The methodology encompasses a comprehensive review of existing RAG implementations in transportation, followed by the design and analysis of a prototype system specifically tailored to railway needs, utilizing domain-specific datasets and natural language queries. This study offers valuable insights into the feasibility and scalability of RAG systems for enhancing the efficiency and responsiveness of railway operations.**



***Keywords—Retrieval-Augmented Generation, Technical Study, Public Sector***


## I. INTRODUCTION

The rapid evolution of Generative AI (GenAI) and highly intelligent large language models (LLMs) has revolutionized information retrieval, with Retrieval-Augmented Generation (RAG) emerging as a next-generation solution. In the public sector, where vast datasets are crucial for decision-making and service delivery, RAG systems combine retrieval-based and generative models to provide accurate, contextually aware responses. With platforms like OpenAI reaching over 100 million active users within months of launch, the growing reliance on AI-driven tools is evident. This study addresses the inefficiencies in traditional public sector information retrieval, which often leads to delays and lack of personalization. By exploring the potential of RAG to enhance accuracy, efficiency, and user experience across domains such as healthcare, education, and government services, the paper aims to propose a framework for integrating RAG systems into public sector operations to improve service delivery and decision-making.

## II. RELATED WORK

In [4] the study highlights the logistical and financial decline of Indian Railways, emphasizing problems like a poor Operating Ratio (OR), Cost of Revenue (COR), and Return on Logistics Assets Ratio (ROLAR), signaling a need for deeper analysis of these issues.

In [7] the paper identifies common failure points when engineering RAG systems in diverse domains (research, education, and biomedical), stressing the challenges and necessary refinements for robust system performance.

In [1] the research introduces the RAG model, utilizing pre-trained seq2seq models (parametric memory) and dense vector indexing of Wikipedia (non-parametric memory) for knowledge-intensive NLP tasks.

In [8] the paper focuses on fine-tuning RAG architectures, particularly the DPR retriever, while addressing the engineering complexities that arise in end-to-end tuning for effective question-answering.

In [9] the research demonstrates the superior performance of RAG-based agents over existing LLMs in science-based QA benchmarks, underscoring its efficacy in scientific research applications.

In [10] the paper offers a comprehensive review of the advancements in RAG systems, exploring critical components, evaluation frameworks, and cutting-edge technologies.

In [11] the paper introduces the FlashRAG toolkit, which supports modular and efficient RAG research through 12 advanced RAG methods, a rich collection of datasets, and comprehensive evaluation metrics.

## III. LIMITATIONS IN EXISTING SYSTEM

1. Context Management for Long Documents

Existing RAG systems may struggle with efficiently managing long contexts, which is crucial for handling detailed documentation, reports, and real-time updates across various public services.

2. Robustness to Misinformation

If a RAG system retrieves outdated or incorrect information, it could lead to inaccurate responses for citizens seeking assistance with services such as healthcare, social welfare, or public safety.

3. Integration of Domain-Specific Knowledge

Current RAG systems may not be well-adapted to the specific needs of different public sector agencies, such as handling diverse and localized information or integrating with various live data sources relevant to specific public services.

4. Resource-Intensive Operations

High computational costs and memory requirements could hinder the feasibility of deploying RAG systems at scale for real-time updates and queries in public sector operations, especially in resource-constrained environments.

5. Generalization and Domain Adaptation

Existing RAG methods do not perform optimally across different public sector domains, affecting the system's ability to effectively handle the varied queries and requirements of citizens interacting with government services.

This gap underscores the need for further exploration to develop effective frameworks for RAG deployment in public services.

## IV. RETRIEVAL AUGMENTED GENERATION

Retrieval-Augmented Generation (RAG) is an innovative framework that combines the strengths of two fundamental components: a retrieval system and a generation system. This hybrid approach enhances the ability to provide accurate and contextually relevant responses to complex queries, making it particularly valuable in various applications, including public sector operations.

1. Retrieval System
   The retrieval component is responsible for sourcing relevant information from a predefined database or knowledge base. It utilizes techniques such as keyword matching, semantic search, and information retrieval algorithms to identify documents or data snippets that relate closely to the user's query. By retrieving pertinent information, this system lays the groundwork for generating informed responses. The effectiveness of the retrieval system is crucial, as it ensures that the generative model operates with accurate and relevant content.
2. Generation System
   The generation component leverages advanced Natural Language Processing (NLP) techniques, often powered by large language models (LLMs), to create coherent and contextually appropriate responses. Once the retrieval system has identified relevant information, the generation model synthesizes this data, generating responses that are not only informative but also human-like in their structure and tone. This component enhances the user experience by providing answers that feel natural and engaging, while also incorporating the specific details retrieved from the database.

Together, these components form a robust RAG system that effectively addresses the challenges of information retrieval and response generation, making it a powerful tool for improving interactions in various domains, including healthcare, education, and public sector services. By integrating retrieval and generation capabilities, RAG systems can offer users more personalized and accurate information, enhancing decision-making and overall service delivery.

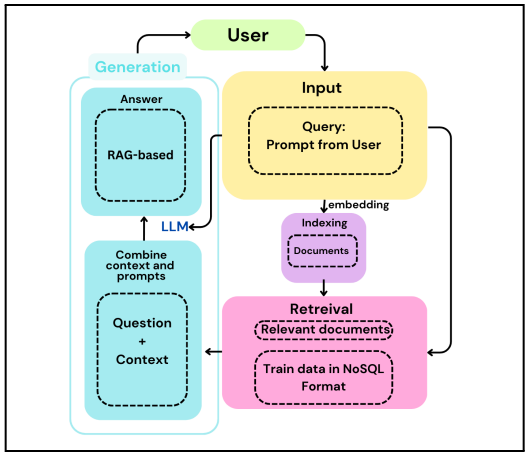



Fig 1 Block Diagram of RAG

**User Query:** The initial input provided by the user.

**Generation Answer:** A preliminary response generated by the language model (LLM) based on internal knowledge.

**Embedding:** Converts the user query into an embedding format suitable for retrieval.

**Indexing:** Organizes and stores documents, typically in NoSQL format, for efficient access.

**Retrieval:** Searches the indexed documents to identify relevant ones based on the embedded query.

**Relevant Documents:** The documents identified as containing pertinent information.

**Combine Context and Prompts:** Integrates the relevant documents with the original query and context.

**RAG-based Generation:** The LLM processes the enriched query to generate a final, contextually informed response.

Output: The final, accurate answer is delivered to the user.

## V. ALGORITHM AND PROCESS DESIGN

Retrieval-Augmented Generation (RAG) system, algorithm and process design are crucial for transforming user queries into meaningful outputs. This involves a series of carefully crafted steps—from converting user input into embeddings, retrieving relevant data from structured sources, to generating natural language responses using advanced language models like GPT-2. Each stage must be meticulously designed to ensure accurate data retrieval, efficient processing, and coherent response generation**.**

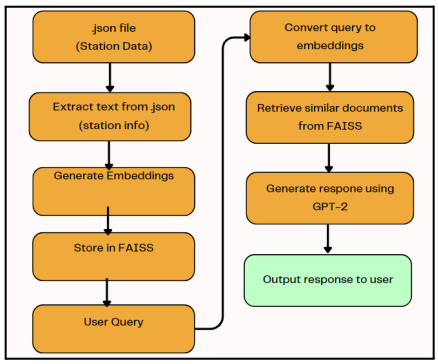



Fig 2 Block Diagram

The integration of a RAG system begins with a JSON file containing station data, which serves as the primary source of structured information regarding various stations. When a user submits a query, the system converts the query into embeddings, transforming the user input into a vector representation that facilitates efficient comparison with existing data. Next, the system extracts relevant text from the JSON file, retrieving pertinent station information that matches the user's query. Utilizing the FAISS (Facebook AI Similarity Search) framework, it then retrieves similar documents, searching the vector database to identify entries closely aligned with the user's input. Following this, the system generates embeddings for the extracted station data, creating vector representations that can be compared with the query embeddings. The heart of the process lies in the response generation using GPT-2, where the model leverages the retrieved data to formulate a coherent and contextually relevant natural language response. Additionally, to optimize future interactions, the system stores the embeddings of the station data in FAISS, ensuring quick access for subsequent queries. Finally, the system outputs the response to the user, presenting the generated answer based on their original inquiry, thereby enhancing the overall user experience.

## VII. IMPLEMENTATION DETAILS

In the rapidly evolving landscape of artificial intelligence (AI), various platforms offer unique capabilities that cater to different user needs and applications. This comparison focuses on three prominent AI platforms: OpenAI, Google Generative AI, and Hugging Face. Each platform presents distinct features, advantages, and limitations that impact their usability and effectiveness in various contexts, particularly in the realm of natural language processing (NLP) and machine learning.

### 1. OpenAI

OpenAI is renowned for its sophisticated language models, particularly the GPT series, which are adept at understanding and generating human-like text. These models can be applied across various domains, including chatbots, content generation, and more. OpenAI offers a comprehensive API that enables developers to seamlessly incorporate these models into their applications.

Disadvantages:

Cost: Utilizing OpenAI's API can be expensive, especially for projects that demand high interaction levels or large volumes of queries.

Usage Limits: The free tier comes with stringent usage restrictions, which may limit experimentation and development for students or smaller initiatives.

Complexity: While the models are powerful, the intricacies involved in their usage may pose challenges for those who are new to the technology without significant technical expertise.

### 2. Google Generative AI

Google provides a suite of Generative AI models tailored for tasks

like content creation and code generation. These models benefit from Google's vast data and advanced machine learning infrastructure, yielding high-quality results across a variety of applications.

Disadvantages:
Cost: Similar to OpenAI, using Google's models can lead to considerable expenses, particularly in business environments.
Usage Limits: Google also places restrictions on free usage, which could limit the capacity to conduct extensive testing or projects for students.
Integration Challenges: Implementing and integrating Google's AI solutions can be intricate, often requiring a deeper understanding of their ecosystem.

### 3. Hugging Face

Hugging Face is an open-source platform that offers a wide range of pre-trained models for tasks in natural language processing and beyond. The Hugging Face Transformers library allows users to easily access and customize these models, making it appealing to both novices and seasoned users.
Reasons for Choosing Hugging Face:

1. Cost-Effective: Models on Hugging Face are free to use, making them accessible for students and those with limited budgets.
2. Open Source: The platform fosters collaboration and resource sharing, enabling students to engage with a lively community and share their discoveries.
3. Flexibility: Users can fine-tune models for specific applications without incurring significant costs, promoting extensive experimentation and learning.

| Pricing | Paid | Paid | Free |
|---|---|---|---|
| Quality | Best quality | Slightly lower | Lower quality |
| Conversational Capability | Best conversational capability | Moderate conversational capability | Little to no conversational capability |
| Token Limit | 4000 Tokens Limit | 24000 Tokens Limit | Can be manually set |
| Quota Impact | Working solution not implemented because of reaching the quota limit | Working solution not implemented because of reaching the quota limit | Working preliminary solution is implemented using 'gpt2' model |

Fig.3 Comparison of language models

**Algorithms Used**

**1.** Recommendation Algorithms

1. Collaborative Filtering:

User-Based Collaborative Filtering:

```
function user_based_collaborative_filtering(user_id):
    similar_users = find_similar_users(user_id)
    recommended_items = []
    for user in similar_users:
        for item in user.recommended_items:
            if item not in user.past_items:
                recommended_items.append(item)
    return unique(recommended_items)
```

Item-Based Collaborative Filtering:

```
function item_based_collaborative_filtering(item_id):
    similar_items = find_similar_items(item_id)
    return user_ratings_for(similar_items)
```

2. Content-Based Filtering:

TF-IDF (Term Frequency-Inverse Document Frequency):

```
function calculate_tfidf(document, corpus):
    tf = term_frequency(document)
    idf = inverse_document_frequency(corpus)
    tfidf = tf * idf
    return tfidf
```

Cosine Similarity:

```
function cosine_similarity(vector_a, vector_b):
    dot_product = sum(a * b for a, b in zip(vector_a,
vector_b))
    magnitude_a = sqrt(sum(a * a for a in vector_a))
    magnitude_b = sqrt(sum(b * b for b in vector_b))
    if magnitude_a == 0 or magnitude_b == 0:
        return 0
    return dot_product / (magnitude_a * magnitude_b)
```

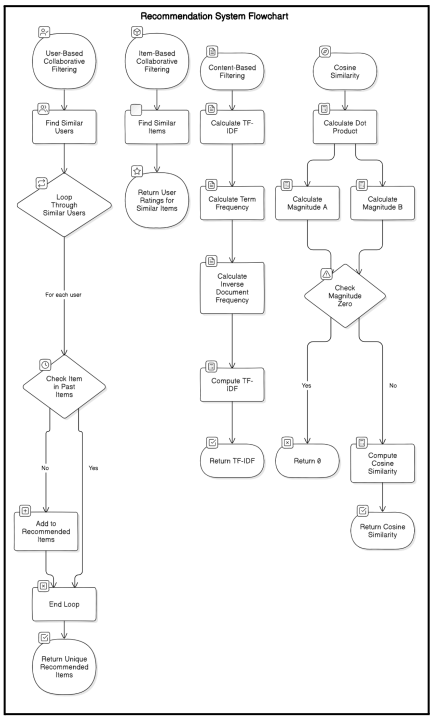



*Fig 3.1 Recommendation System Flowchart*

**Machine Learning Algorithms**

Purpose: These algorithms predict train delays and user behavior based on historical data.

1. Regression Analysis:

Linear Regression:

```
function linear_regression(training_data):
    coefficients = initialize_coefficients()
    for epoch in range(num_epochs):
        for data_point in training_data:
            prediction = predict(data_point.features,
coefficients)
            error = prediction - data_point.target
            update_coefficients(coefficients, error,
data_point.features)
    return coefficients
```

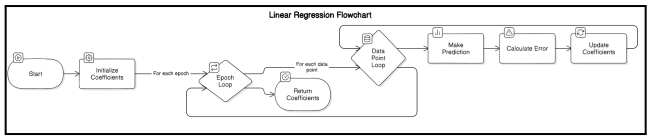



*Fig 4.1 Linear Regression Algorithm*

Logistic Regression:

```
function logistic_regression(training_data):
    coefficients = initialize_coefficients()
    for epoch in range(num_epochs):
        for data_point in training_data:
            prediction = sigmoid(predict(data_point.features,
coefficients))
            error = prediction - data_point.target
            update_coefficients(coefficients, error,
data_point.features)
    return coefficients
```

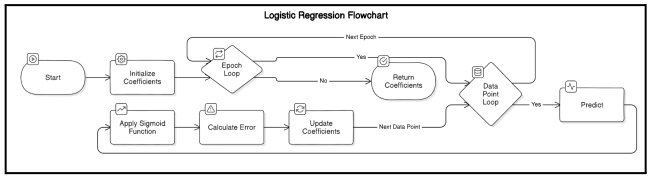



Fig 4.2 Logistic Regression Algorithm

2. Decision Trees:

CART (Classification and Regression Trees):

```
function cart(training_data):
    if stopping_condition_met(training_data):
        return create_leaf_node(training_data)
    best_split = find_best_split(training_data)
    left_data, right_data = split_data(training_data, best_split)
    node = create_internal_node(best_split)
    node.left = cart(left_data)
    node.right = cart(right_data)
    return node
```

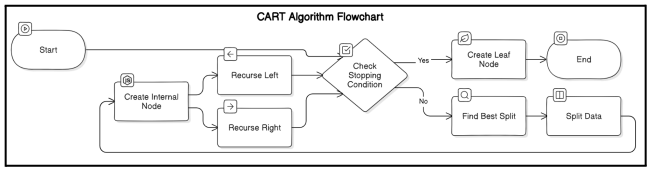



Fig 4.3 CART Algorithm

**Natural Language Processing (NLP) Algorithms**

Tokenization:

```
function tokenize(text):
    return text.split() // split text by whitespace
```

Named Entity Recognition (NER):

```
function named_entity_recognition(text):
    entities = []
    for word in tokenize(text):
        if is_entity(word):
            entities.append(word)
    return entities
```

Sentiment Analysis:

```
function sentiment_analysis(text):
    score = 0
    for word in tokenize(text):
        score += sentiment_score(word)
    if score > 0:
        return "Positive"
    else if score < 0:
        return "Negative"
    else: return "Neutral"
```

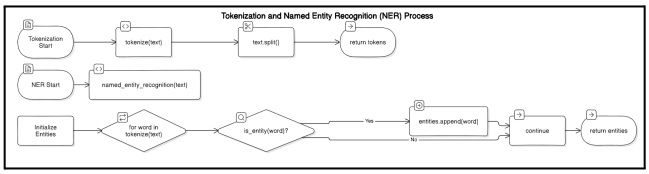



*Fig 21.4 Tokenization and NER*

## VIII. RESULTS AND DISCUSSION

When querying railway stations within the KR railway zone, the system successfully identified several key stations, including Kumta (KT), Veer (VEER), and Kudal (KUDL). Other notable stations within the same zone include Khed (KHED), Kalambani (KLMB), and Indapur (INP). Additionally, stations such as Kamthe (KMAH), Kharepatan (KRPM), and Kankavli (KKW) were also recognized as part of the KR zone.

In a separate inquiry regarding the station 'Roha' (ROHA), the system provided clear and concise information indicating that it is part of the Central Railway (CR) zone. These results highlight the effectiveness of the system in retrieving accurate and relevant data based on user queries, showcasing its potential to assist users in navigating railway services efficiently. The integration of such features not only enhances user experience but also promotes accessibility to vital transportation information across different railway zones.

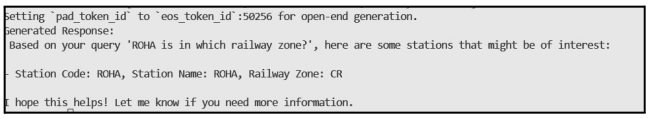



Figure 3.1


Figure 3.2

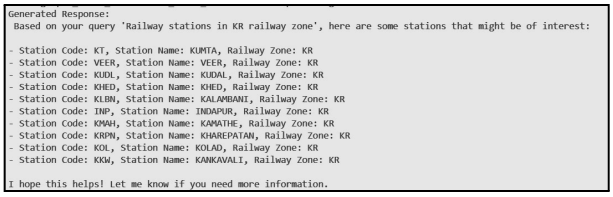



Figure 3.3


Figure 3.4

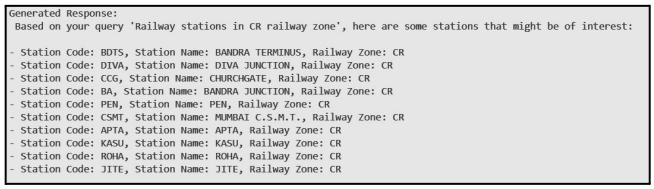



Figure 3.4

## VIII. CONCLUSION

The development and implementation of a Retrieval Augmented Generation (RAG) system for the railway sector represents a significant advancement in how data is managed, analyzed, and reported. By combining sophisticated retrieval techniques with powerful generative language models, this project addresses critical challenges faced by the railway industry in handling large volumes of diverse data.

By integrating advanced retrieval and generative capabilities, the RAG system not only enhances the efficiency of data processing but also supports a more proactive and responsive approach to managing railway operations. This innovative solution is poised to transform how the railway sector leverages data, ultimately contributing to more effective and efficient railway operations and improved service quality.

## IX. FUTURE WORK

In order for any organization /public sector company to go forward and implement this RAG,while requires several steps to follow.

1. Define Objectives

Goal Clarification: Establish clear objectives for integrating the chosen Hugging Face model into the project, such as enhancing response accuracy, increasing user engagement, or automating specific tasks.

2. Research and Preparation

Literature Review: Perform a thorough review of existing literature on Retrieval-Augmented Generation (RAG) systems and their applications within the public sector to guide the development process.

3. Data Collection and Preparation

Data Acquisition: Gather pertinent datasets for training and fine-tuning the model.
Data Processing: Clean and preprocess the data to ensure its quality, which includes removing duplicates, addressing missing values, and proper formatting.

4. Model Development

Fine-Tuning: Adapt the selected Hugging Face model(s) by fine-tuning them with the prepared dataset to fit the specific project context.
Integration Framework: Create a framework for incorporating the model into existing public sector systems or applications.

5. Testing and Validation

Performance Evaluation: Conduct tests to assess the model's accuracy, efficiency, and response times.
User Input: Collect feedback from potential users to identify opportunities for enhancement.

6. Implementation

Deployment: Launch the integrated system in a controlled setting to allow for real-world testing and necessary adjustments.
Monitoring Systems: Set up mechanisms to track the model's performance and user interactions.

7. Evaluation and Refinement

Result Analysis: Analyze data from the deployment phase, focusing on metrics like user engagement, accuracy, and overall system performance.
Continuous Improvement: Utilize findings to make iterative enhancements to both the model and the integration process.

8. Documentation and Reporting

Comprehensive Documentation: Create detailed documentation that outlines the development process, challenges encountered, and implemented solutions.
Final Report: Compile a final report that summarizes project outcomes, lessons learned, and recommendations for future initiatives.

9. Presentation

Stakeholder Presentation: Develop a presentation to communicate findings to stakeholders, emphasizing the benefits and impacts of integrating the RAG system into public sector operations.

## X. REFERENCES

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", 22 May 2020 (v1), 12 Apr 2021 (this version, v4)

[2] Isamu Isozaki, “Literature Review on RAG(Retrieval Augmented Generation) for Custom Domains”, Nov 26, 2023

[3] Kieran Pichai, “A Retrieval-Augmented Generation Based Large Language Model Benchmarked On a Novel Dataset”, November 2023, Journal of Student Research,
DOI:10.47611/jsrhs.v12i4.6213, License :CC BY-NC-SA 4.0

[4] Shouvik Sanyal, Alam Ahmad, Hafiz Wasim Akram, “An Analysis of Performance of Indian Railways”, January 2021, DOI:10.1504/IJLSM.2021.10043738

[5] Mohd Arshad, Muqeem Ahmed, “Prediction of Train Delay in Indian Railways through Machine Learning Techniques ”, February 2019, International Journal of Computer Sciences and Engineering 7(2):405-4117(2):405-411,
DOI:10.26438/ijcse/v7i2.405411.

[6] Mohd Arshad, Muqeem Ahmed, “Train Delay Estimation in Indian Railways by Including Weather Factors Through Machine Learning Techniques”, September 2019, DOI:10.2174/2666255813666190912095739.

[7]Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, Mohamed Abdelrazek, “Seven Failure Points When Engineering a Retrieval Augmented Generation System”, 2024 IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN), 18 June, 2024

[8]Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Suranga Nanayakkara, “FINE-TUNE THE ENTIRE RAG ARCHITECTURE (INCLUDING DPR RETRIEVER) FOR QUESTION-ANSWERING”, arXiv:2106.11517v1, 23 June, 2021

[9] Jakub Lala  Odhran O’Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G Rodriques, Andrew D White, “PaperQA: Retrieval-Augmented Generative Agent for Scientific Research”, arXiv:2312.07559v2 , 14 December, 2023

[10]Yunfan Gaoa , Yun Xiongb , Xinyu Gaob , Kangxiang Jiab , Jinliu Panb , Yuxi Bic , Yi Daia , Jiawei Suna , Meng Wangc , and Haofen Wang, “Retrieval-Augmented Generation for Large Language Models: A Survey”, arXiv:2312.10997v5 [cs.CL] , 27 March, 2024

[11]Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, Zhicheng Dou,”FlashRAG: Modular Toolkit for Efficient Retrieval-Augmented Generation Research”, arXiv:2405.13576v1 [cs.CL], 22 May, 2024

[12]Chidaksh Ravuru, Sagar Srinivas Sakhinana, Venkataramana Runkana, “Agentic Retrieval-Augmented Generation for Time Series Analysis”, arXiv:2408.144846v1 [cs.CL], 18 Aug, 2024

[13]Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, Zhicheng Dou, “MemoRAG: Moving Towards Next-Gen RAG via Memory-Inspired Knowledge Discovery”, arXiv:2409.055916v2 [cs.CL], 10 Sept, 2024

[14]Zahra Sepasdar, Sushant Gautam, Cise Midoglu, Michael A. Riegler, Pål Halvorsen, “Enhancing Structured-Data Retrieval with GraphRAG: Soccer Data Case Study”, arXiv:2409.17580v1 [cs.IR], 26 Sept, 2024

[15]Jordi Bayarri-Planas, Ashwin Kumar Gururajan, Dario Garcia-Gasulla, “Boosting Healthcare LLMs Through Retrieved Context”, arXiv:2409.13127v1 [cs.AI], 23 Sept, 2024

[16] Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, Manaal Faruqui, “Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation”, arXiv:2409.12941 [cs.CL], 19 Sept, 2024

[17] Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, Bryan Catanzaro, “RankRAG: Unifying Context Ranking with Retrieval-Augmented Generation in LLMs”, arXiv:2407.02485v1 [cs.CL], 19 Sept, 2024

## 2. Plagiarism report of the paper draft

RAG Systems in Railways

ORIGINALITY REPORT

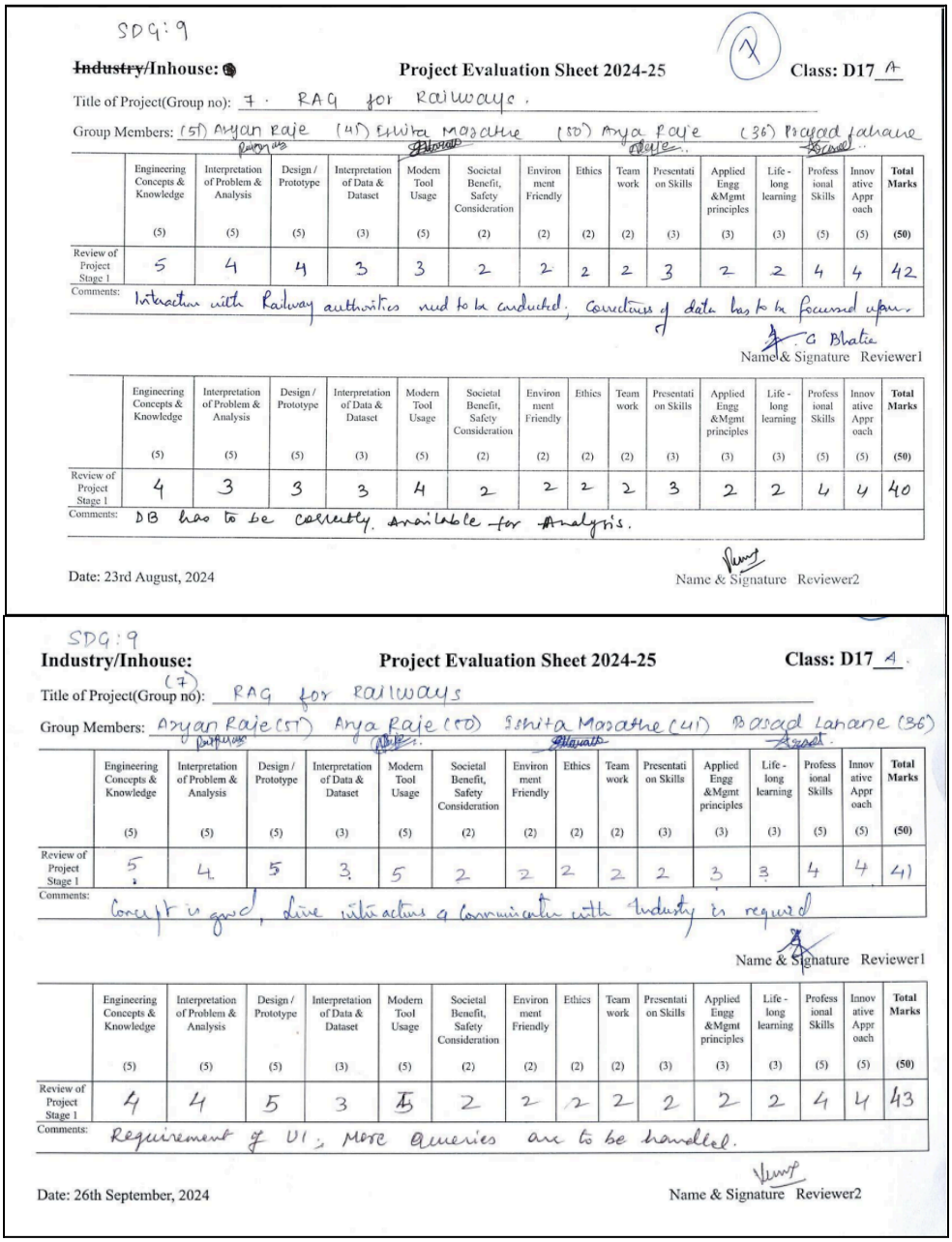
| 5% | 3% | 2% | 4% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | vesit.ves.ac.in<br>Internet Source | 3% |
| 2 | Submitted to Johns Hopkins Unversity<br>Student Paper | 1% |
| 3 | Submitted to City University<br>Student Paper | 1% |

Exclude quotes On
Exclude bibliography On
Exclude matches < 1%

## 3. Xerox of project review sheet



SDG: 9

~~Industry~~/Inhouse:  **Project Evaluation Sheet 2024-25**  Class: D17_A

Title of Project(Group no): 7. RAG for Railways.

Group Members: (51) Aryan Raje (41) Ishita Masathe (50) Arya Raje (36) Prasad Lahane

| | Engineering Concepts & Knowledge (5) | Interpretation of Problem & Analysis (5) | Design / Prototype (5) | Interpretation of Data & Dataset (3) | Modern Tool Usage (5) | Societal Benefit, Safety Consideration (2) | Environment Friendly (2) | Ethics (2) | Team work (2) | Presentation Skills (3) | Applied Engg &Mgmt principles (3) | Life-long learning (3) | Professional Skills (5) | Innovative Approach (5) | Total Marks (50) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Review of Project Stage 1 | 5 | 4 | 4 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 4 | 4 | 42 |

Comments: Interaction with Railway authorities need to be conducted. Correctness of data has to be focussed upon.

G Bhatia
Name & Signature Reviewer1

| | Engineering Concepts & Knowledge (5) | Interpretation of Problem & Analysis (5) | Design / Prototype (5) | Interpretation of Data & Dataset (3) | Modern Tool Usage (5) | Societal Benefit, Safety Consideration (2) | Environment Friendly (2) | Ethics (2) | Team work (2) | Presentation Skills (3) | Applied Engg &Mgmt principles (3) | Life-long learning (3) | Professional Skills (5) | Innovative Approach (5) | Total Marks (50) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Review of Project Stage 1 | 4 | 3 | 3 | 3 | 4 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 4 | 4 | 40 |

Comments: DB has to be correctly available for Analysis.

Date: 23rd August, 2024
Name & Signature Reviewer2

SDG: 9

**Industry/Inhouse:**  **Project Evaluation Sheet 2024-25**  **Class: D17_A**

Title of Project(Group no): (7) RAG for Railways

Group Members: Aryan Raje (51) Arya Raje (50) Ishita Masathe (41) Prasad Lahane (36)

| | Engineering Concepts & Knowledge (5) | Interpretation of Problem & Analysis (5) | Design / Prototype (5) | Interpretation of Data & Dataset (3) | Modern Tool Usage (5) | Societal Benefit, Safety Consideration (2) | Environment Friendly (2) | Ethics (2) | Team work (2) | Presentation Skills (3) | Applied Engg &Mgmt principles (3) | Life-long learning (3) | Professional Skills (5) | Innovative Approach (5) | Total Marks (50) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Review of Project Stage 1 | 5 | 4 | 5 | 3 | 5 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 4 | 4 | 41 |

Comments: Concept is good, Live interactions & communication with Industry is required

Name & Signature Reviewer1

| | Engineering Concepts & Knowledge (5) | Interpretation of Problem & Analysis (5) | Design / Prototype (5) | Interpretation of Data & Dataset (3) | Modern Tool Usage (5) | Societal Benefit, Safety Consideration (2) | Environment Friendly (2) | Ethics (2) | Team work (2) | Presentation Skills (3) | Applied Engg &Mgmt principles (3) | Life-long learning (3) | Professional Skills (5) | Innovative Approach (5) | Total Marks (50) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Review of Project Stage 1 | 4 | 4 | 5 | 3 | 5 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 4 | 4 | 43 |

Comments: Requirement of UI; More queries are to be handled.

Date: 26th September, 2024
Name & Signature Reviewer2