

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

**An Autonomous Institute Affiliated to University of Mumbai
Department of Computer Engineering**



Project Report on

RAG for Railways

In partial fulfilment of the Fourth Year, Bachelor of Engineering (B.E.) Degree in
Computer Engineering at the University of Mumbai
Academic Year 2024-25

Submitted by

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

Project Mentor
Dr. Mrs. Gresha Bhatia

(2024-25)

**VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF
TECHNOLOGY**
Department of Computer Engineering



Certificate

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 have satisfactorily completed the project on "**RAG for Railways**" as a part of their coursework of PROJECT-II for Semester-VIII under the guidance of their mentor **Dr. Mrs. Gresha Bhatia** in the year 2024-25 .

This project report entitled **RAG for Railways** by *Aryan Raje, Arya Raje, Ishita Marathe, Prasad Lahane* is approved for the degree of **B.E. Computer Engineering**.

Programme Outcomes	Grade
PO1,PO2,PO3,PO4,PO5,PO6,PO7, PO8, PO9, PO10, PO11, PO12 PSO1, PSO2	

Date:

Project Guide:

Project Report Approval

For

B. E (Computer Engineering)

This project report entitled **RAG for Railways** by *Aryan Raje, Arya Raje, Ishita Marathe, Prasad Lahane* is approved for the degree of **B.E. Computer Engineering**.

Internal Examiner

External Examiner

Head of the Department

Principal

Date:

Place: Mumbai

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Aryan Raje(51)

Arya Raje(50)

Ishita Marathe(41)

Prasad Lahane(36)

Date:

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	Able to apply the relevant engineering concepts, knowledge and skills towards the project.
CO2	Able to identify, formulate and interpret the various relevant research papers and to determine the problem.
CO 3	Able to apply the engineering concepts towards designing solutions for the problem.
CO 4	Able to interpret the data and datasets to be utilised.
CO 5	Able to create, select and apply appropriate technologies, techniques, resources and tools for the project.
CO 6	Able to apply ethical, professional policies and principles towards societal, environmental, safety and cultural benefit.
CO 7	Able to function effectively as an individual, and as a member of a team, allocating roles with clear lines of responsibility and accountability.
CO 8	Able to write effective reports, design documents and make effective presentations.
CO 9	Able to apply engineering and management principles to the project as a team member.
CO 10	Able to apply the project domain knowledge to sharpen one's competency.
CO 11	Able to develop a professional, presentational, balanced and structured approach towards project development.
CO 12	Able to adopt skills, languages, environment and platforms for creating innovative solutions for the project.

Index

Chapter No.	Title	Page No.
1	Introduction	12
1.1	Introduction to the project	12
1.2	Motivation for the project	13
1.3	Problem Definition	14
1.4	Existing Systems	15
1.5	Lacuna of the Existing Systems	15
1.6	Relevance of the Project	16
2	Literature Survey	17
A	Brief overview of Literature Survey	17
2.1	Research Papers a. Abstract of the research paper b. Inference drawn from the paper	17
2.2	Articles Referred	22
2.3	Interaction with domain experts	23
2.4	Comparison with Existing Systems	24
3.	Requirement Gathering for the proposed System	25
3.1	Introduction to Requirement Gathering	25
3.2	Functional Requirements	26
3.3	Non-Functional Requirements	27
3.4	Hardware, Software, Technology and tools utilised	27
3.5	Constraints	28
4.	Proposed Design	30
4.1	Block diagram of the system	30
4.2	Modular design of the system	33
4.3	Detailed Design	33
4.4	Project Scheduling and Timeline Charts	36
5.	Implementation of the Proposed System	37
5.1	Methodology employed for development	37

5.2	Algorithms and flowcharts for the respective modules developed	38
5.3	Datasets source and utilisation	42
6.	Testing of the Proposed System	44
6.1	Introduction to testing	44
6.2	Types of tests Considered	45
6.3	Various test case scenarios considered	45
6.4	Inference drawn from the test cases	46
7.	Results and Discussions	41
7.1	Screenshots of User Interface (UI) for the respective module	48
7.2	Performance Evaluation measures	50
7.3	Input Parameters / Features considered	51
7.4	Comparison of results with existing systems	53
7.5	Inference drawn	53
8.	Conclusion	54
8.1	Limitations	54
8.2	Conclusion	54
8.3	Future Scope	54
	References	55
	Appendix	57
a	Paper I	57
b	Plagiarism Report of Paper I	64
c	Paper II	65
d	Plagiarism Report of Paper II	72
e	Project review sheet 1	73
f	Project review sheet 2	73

List of Figures:

Fig no.	Heading	Page no.
1	RAG system	12
2	Logo Konkan Railway	13
3	.NTES dashboard	14
4.1	Current Train Position on Konkan Railway-	14
4.2	Chatbot Interface	15
5.1	Cover Page of International initiative for impact innovation	22
5.2	Wikipedia Link	22
5.3	KRCL website	23
6.1	Proposed Email to the CPIO of Konkan Railways.	23
6.2	Reply from Chief Manager (Admn) and CPIO of Konkan Railways - Mr. Balasaheb B Nikam	24
7	Block diagram of proposed system.	30
8	Modular Diagram	31
9.1	Data Flow Diagram Level 0	33
9.2	Data Flow Diagram Level 1	34
9.3	Data Flow Diagram Level 1	34
9.4	Flowchart Of Proposed System	35
10	Gantt chart	36
11	Recommendation System Flowchart	39
12.1	Linear Regression Algorithm	40
12.2	Logistic Regression Algorithm	40
12.3	CART Algorithm	41
12.4	Tokenization and NER	41
13.1	Real-time data extraction	43
13.2	Manual Data Collection	43
14.1	Screenshot of Output 1	48
14.2	Screenshot of output 2	49
14.3	Screenshot of output 3	50
15	Plagiarism Report Paper 1	64
16	Plagiarism Report Paper 2	72

List of Tables:

Table no.	Heading	Page no.
1	Comparison with the existing system	24
2	Requirements of the system	26
3	Test Cases	46
4	Evaluation Measures	50
5	Difference between RAG for Railways and KR website	53

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.

Chapter 1: Introduction

1.1. Introduction:

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.

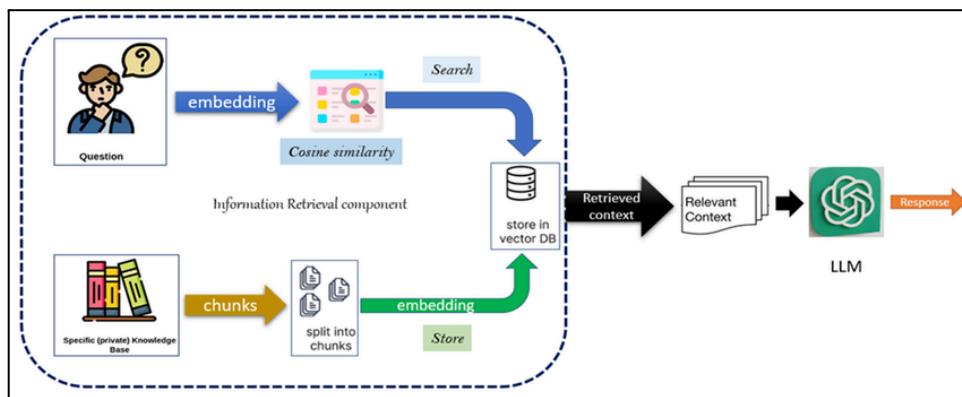


Fig.1. RAG system

(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.

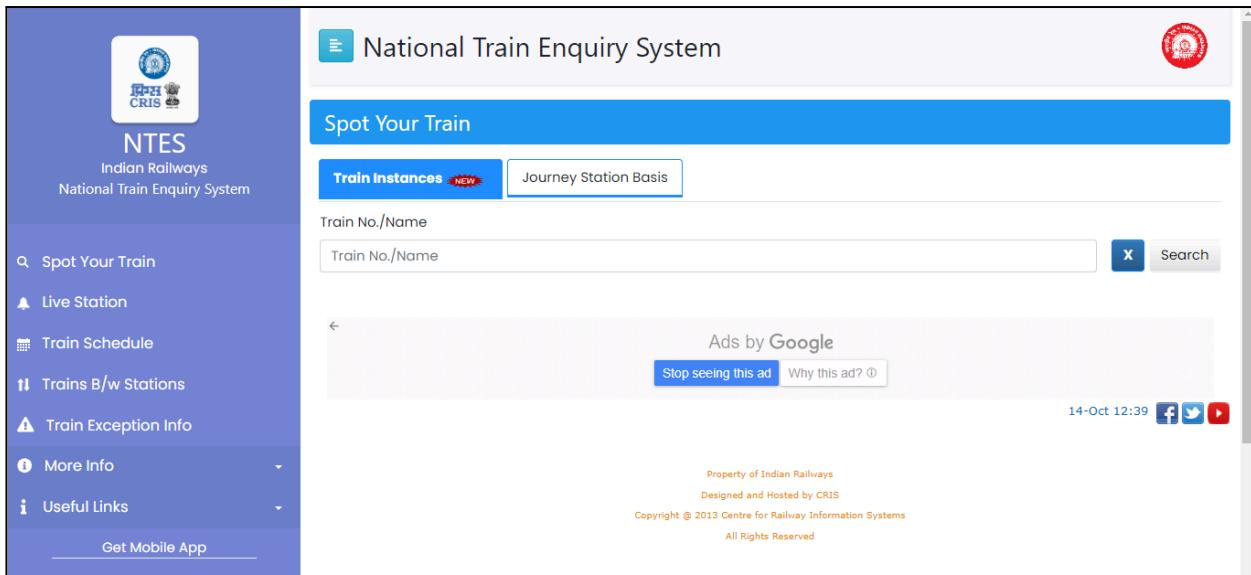


*Fig.2.Logo Konkan Railway
(Source:Konkan Railway.com , irctc.co.in)*

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:

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. 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/>)

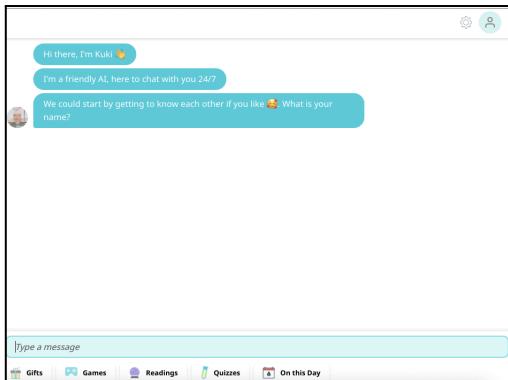


Fig. 4.2. Chatbot Interface

(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.

1.4. Existing Systems:

Existing systems have paid little attention to the skills and jobs as a combined feature. LinkedIn has become the world's largest online professional networking site with more than 562 million members in over 200 countries and territories worldwide but it only connects the recruiter to the people [12]. Coursera is a platform that provides various courses but does not connect the recruiter to the people [13]. Thus a combined solution is required. ALISON (Advance Learning Interactive Systems Online) is an Irish online education platform for higher education, teaching workplace skills and exploring new interests that provides certificate courses and accredited diploma courses.

1.5. Lacuna of the Existing System:

The current system falls short in several ways that make it frustrating and inconvenient for users. For starters, it doesn't understand natural language, so people have to dig through complicated menus just to find basic information. This can be especially tough for tourists or infrequent travelers who aren't familiar with train details. The interface feels rigid and outdated—it doesn't respond to individual needs or offer much flexibility. There's no personalization or smart responses based on what the user might actually be looking for, and it doesn't use any modern tools like AI to simplify or summarize content. Regional networks like the Konkan Railway are often overlooked, which makes things even harder for people traveling in those areas. On top of that, users have to enter station names or codes exactly right, which can lead to mistakes and dead ends. The system also lacks any sort of chatbot or conversational help, so it can't really follow what users want if they ask follow-up questions or express their needs in a more natural way. Finally, it doesn't connect with other helpful info—like nearby attractions or local transport—so it misses the bigger picture of what travelers actually need.

1.6. 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.

Chapter 2: Literature Survey

A. Overview of literature survey:

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. Research Papers :

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.

Inference Drawn:

Indian Railways' financial and operational struggles demand urgent reforms, including subsidy cuts and dynamic pricing, to ensure long-term viability.

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.

Inference Drawn:

RAG systems improve LLM responses but require ongoing validation and iterative refinement to ensure reliability.

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.

Inference Drawn:

RAG models enhance knowledge-intensive NLP tasks by integrating parametric and non-parametric memory for more factual and precise outputs.

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.

Inference Drawn:

End-to-end fine-tuning improves RAG's question-answering performance over the original architecture, with engineering solutions and open-source implementation provided.

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.

Inference Drawn:

RAG-based agent for scientific literature, outperforms existing LLMs and matches human experts in accuracy and synthesis on complex science QA tasks.

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.

Inference Drawn:

RAG systems enhance LLMs by integrating external knowledge to improve accuracy and transparency, with evolving paradigms and evaluation frameworks paving the way for future advancements.

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

Inference Drawn:

The proposed modular, open-source RAG toolkit addresses the lack of standardization by enabling efficient comparison, customization, and evaluation of diverse RAG methods in a unified framework.

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.

Inference Drawn:

FRAMES shows LLMs struggle with complex RAG tasks but improve with multi-step retrieval, underscoring the need for better retrieval-reasoning integration.

Paper 9:

Boosting Healthcare LLMs Through Retrieved Context[15]

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.

Inference Drawn:

Optimized retrieval-augmented open LLMs can match private models in healthcare QA, with OpenMedPrompt enhancing reliability for open-ended medical responses.

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.

Inference Drawn:

Structured-GraphRAG enhances retrieval accuracy and efficiency in complex datasets by leveraging knowledge graphs, outperforming traditional RAG methods while maintaining broad applicability.

Paper 11:

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

Hongjin Qian et.al *arXiv:2409.05591v2 [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.

Inference Drawn:

MemoRAG's dual-system architecture—combining long-term memory with optimized retrieval—outperforms conventional RAG in both complex and straightforward tasks by effectively handling ambiguous queries and unstructured knowledge.

Paper 12:

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

Chidaksh Ravuru et.al *arXiv:2408.14484v1 [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.

Inference Drawn:

The proposed agentic RAG framework with hierarchical multi-agent architecture and prompt retrieval achieves state-of-the-art time series modeling by effectively handling spatio-temporal dependencies and distribution shifts.

2.2 Inference Drawn:

- **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].
- **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].
- **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].
- **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].
- **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].
- **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].
- **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].

2.3 Articles referred

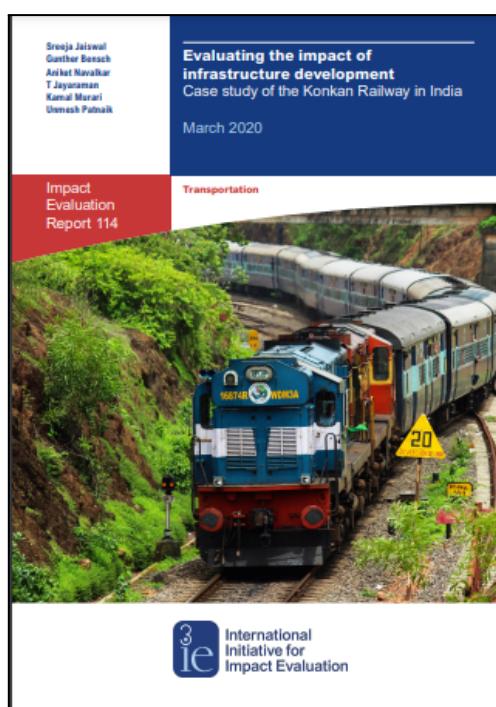
We referred to this article that provided us with more insights into the railway infrastructure.

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.5.1.Cover Page of International initiative for impact innovation
(Source:<https://www.3ieimpact.org>)*

Konkan Railways Wikipedia

To get information about the railway zones, stations etc. whatever static data is necessary.

Konkan Railway zone

The Konkan Railway (abbreviated KR) is **one of the 19 railway zones in India** with its headquarters at CBD Belapur in Navi Mumbai, Maharashtra, India.

Konkan Railway Corporation

Verna, Goa

CBD Belapur

*Fig.5.2.Wikipedia Link
(Source:google.com)*

Konkan Railways Official Website

Regarding any current notifications, alerts, trains schedule

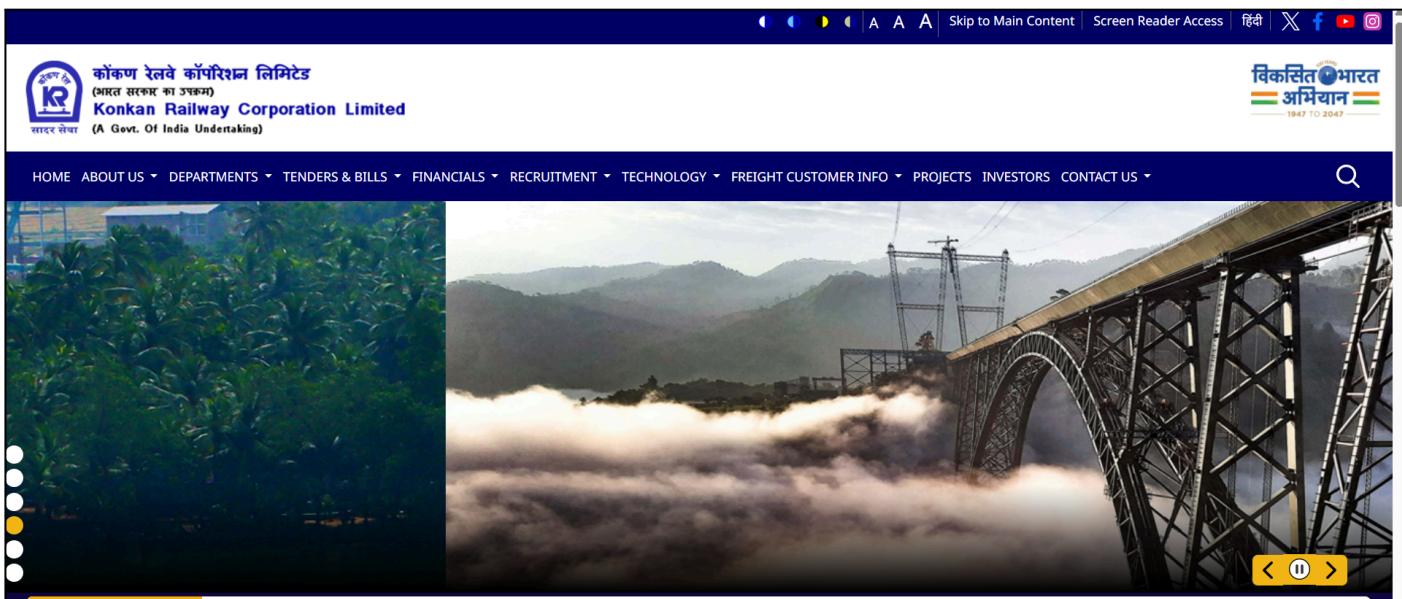


Fig.5..KRCL website
(Source:<https://www.konkanrailway.com/>)

2.4 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.

ARYAN RAJE <d2021.aryan.raje@ves.ac.in>
to VAISHALI, ISHTA, ARYA, PRASAD, gresha.bhatia, BALASAHEB ▾

Respected Ma'am,

We are a team of final-year computer engineering students from Vivekanand Education Society's Institute of Technology, Mumbai, working on a project titled "**RAG for Railways**." The objective of our project is to work on railway management/monitoring by integrating real-time data using advanced technologies such as APIs and artificial intelligence to implement Retrieval Augmented Generation.

We would like to request a meeting with your department to discuss the project's relevance to railway operations, gain valuable insights, and explore potential collaboration opportunities with Konkan Railways. We believe this collaboration could mutually benefit both our academic project and the operational goals of your organization.

We would be highly obliged to schedule a meeting and appreciate the opportunity to present our project at a time convenient for you.

Thank you for considering our request. We look forward to your positive response and the possibility of working together.

PFA
- Letter signed by DHOD Dr. Mrs. Gresha Bhatia (English)
- Letter signed by DHOD Dr. Mrs. Gresha Bhatia (Marathi)

Thanks and Regards,
Aryan Raje d2021.aryan.raje@ves.ac.in
Arya Raje d2021.arya.raje@ves.ac.in
Ishita Marathe 2021.ishita.marathe@ves.ac.in
Prasad Lahane d2021.prasad.lahane@ves.ac.in
+918104982720/+917887412777
VES Institute of Technology, Chembur

Fig 6.1 Proposed Email to the CPIO of Konkan Railways.

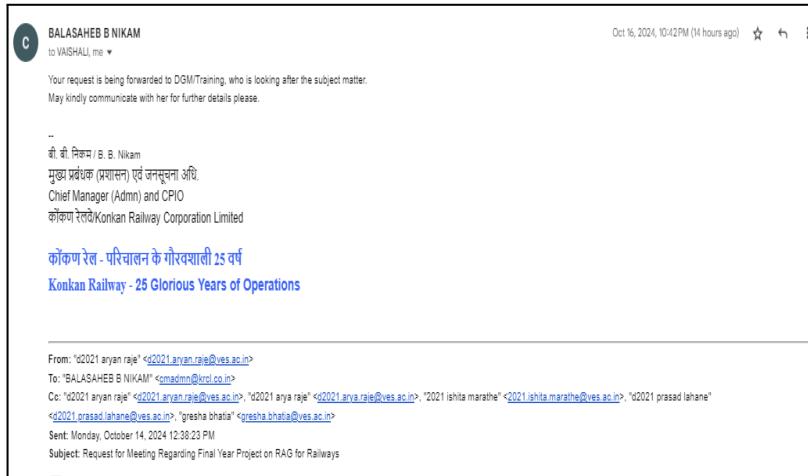


Fig 6.2 Reply from Chief Manager (Admn) and CPIO of Konkan Railways - Mr. Balasaheb B Nikam

2.5 Comparison with the Existing Systems:

Existing Systems	Proposed System
Primarily rely on static databases for railway information.	Integrates real-time data using web scraping and APIs.
Provide keyword-based search results.	Uses a RAG model for contextual, meaningful responses.
Only retrieve documents matching query similarity.	Uses Mistral AI to generate structured, human-like answers.
Lack conversational capabilities and struggle with complex queries.	Handles natural language queries and complex real-time scenarios.
Do not provide guidance on railway navigation.	Helps users find optimal routes and train connections .

Table 1. Comparison with existing system

Chapter 3: Requirement Gathering for the Proposed System

In this chapter we are going to discuss the resources we have used and how we analysed what the user actually needs and what we can provide. We will also discuss the functional and non-functional requirements and finally the software and hardware used.

3.1. Introduction to Requirement Gathering:

The Requirement Gathering is a process of requirements discovery or generating list of requirements or collecting as many requirements as possible by end users. It is also called as requirements elicitation or requirement capture.

The requirements gathering process consists of six steps :

- Identify the relevant stakeholders
- Establish project goals and objectives
- Elicit requirements from stakeholders
- Document the requirements
- Confirm the requirements
- Prioritise the requirements

Use Case	Description
User Query Processing	The system processes user queries related to railway stations, train schedules, and live status updates.
Retrieve Railway Zone Information	Users can ask about a station's railway zone, and the system provides the station code, name, and zone details.
List Stations in a Railway Zone	Users can request a list of all stations within a specific railway zone, which will be retrieved from the database.
Real-Time Train Status	The system fetches real-time train status, including train number, last crossed station, time, and delay updates.
Complex Query Handling	The system, enhanced with Mistral AI, interprets complex queries and provides structured responses beyond simple document retrieval.
Similarity Search with Hugging Face	The system uses Hugging Face embeddings to retrieve relevant documents based on query similarity.
Enhanced Responses with Mistral AI	Mistral AI improves response quality by generating human-like and logically structured answers.

Conversational Interaction	Unlike traditional search systems, the RAG model can handle conversational queries naturally.
Integration with External Data Sources	The system can fetch job openings, real-time train data, and updates from external APIs like Indian Railways and social media sources.
Personalized Recommendations	The system can suggest relevant stations, trains, and routes based on user history and preferences.
User Feedback Mechanism	Users can provide feedback on responses to refine and improve the model over time.

Table No: 2 Requirements of the system

3.2. Functional Requirements:

In our system, we have developed a web-based portal specifically designed to assist Indian Railways commuters by integrating retrieval-augmented generation (RAG) with Mistral AI for context-aware, intelligent responses. This system allows users to query railway-related information in natural language and receive structured, accurate responses.

Key functionalities include:

- **Station and Zone Identification:** Users can query about a station's railway zone and receive structured responses containing *Station Code, Station Name, and Railway Zone*.
- **Zone-wise Station Listings:** Queries like “List all stations in KR Railway Zone” return comprehensive details of **all stations** in the respective zone.
- **Train Status Retrieval:** Users can inquire about the real-time status of a train (e.g., “*What is the time status of MAHARASHTRA SPLENDOUR?*”), and the system fetches the top 3-5 *relevant documents* from the train status dataset, ensuring the most *up-to-date and relevant* information.
- **Contextual Understanding of Complex Queries:** Unlike traditional keyword-based systems, our Mistral AI integration allows the system to reason and provide structured responses. For instance, if a user asks about traveling to Roha, the system intelligently suggests *boarding the “LTT-ROHA PASSENGER” from Thane*, rather than simply retrieving unrelated documents.
- **Real-Time Train Tracking:** The system fetches live train status, including *train number, last crossed station, arrival time, and delay status*, ensuring that commuters get real-time updates efficiently.
- **Conversational Interaction:** Unlike conventional search engines, the system provides human-like responses to general queries such as “*Hello, how are you?*”, making the interaction more engaging.

3.3 Non-Functional Requirements:

- **User-Friendly Interface:** The system ensures a simple and intuitive user interface, making navigation and query interaction seamless.
- **High Data Accuracy:** The dataset used for retrieval and prediction is highly accurate, ensuring reliable railway information.
- **Optimized Dashboard:** The dashboard dynamically updates with real-time train status, zone information, and commuter recommendations.
- **Improved Model Accuracy:** The RAG-based system is trained to enhance response quality and contextual accuracy, making railway queries more precise.
- **Platform Independence:** The application is designed to run across various platforms and devices, ensuring accessibility for all users.
- **Scalability & Advanced Search:** The system integrates advanced search features, allowing users to filter and access relevant railway data efficiently from extensive databases.

3.4 Hardware, Software, Technology and Tools Utilised:

A. Hardware Requirements:-

Minimum Configuration of Laptop/PC

- Processor- 12th Gen Intel(R) Core(TM) i7-12700 2.10 GHz
- Installed RAM:- 16.0 GB (15.7 GB usable)
- System type 64-bit operating system, x64-based processor
- CPUs/GPUs: 2.4 GHz (Base) - 4.2 GHz (Max) - 4 Cores - 8 Threads - 8 MB Cache
- RAM: 8/16 GB RAM
- Storage: 1 TD + 256 SSD | Cloud Storage

B. Software Requirements:-

- **Python** (v3.12.2)
- **FAISS** (v1.7.4)
- **NumPy** (v1.26.4)
- **Pandas** (v2.2.2)
- **JSON-NoSQL** (versionless; refers to JSON format used in NoSQL databases)
- **Streamlit** (v1.33.0)
- **Mistral AI** (v7B Instruct – latest open-weight model)

Techniques:-

- **Python (v3.12.2):** Python is a high-level, general-purpose programming language. Its design philosophy emphasises code readability with the use of significant indentation.
- **NumPy (v1.26):** NumPy is a powerful numerical computing library used for efficient array operations and mathematical computations. It plays a crucial role in handling vectorized data and performing similarity calculations for job recommendations and resume parsing.
- **Pandas (v2.2):** Pandas is a data manipulation and analysis library that is used to process and manage structured datasets. It helps in cleaning, filtering, and transforming data from job listings, resumes, and user profiles for meaningful insights and recommendations.
- **FAISS (v1.7.4):** FAISS is a highly efficient similarity search library used for fast retrieval of relevant job listings based on skills and resume parsing. It allows the system to quickly search and recommend jobs by comparing embeddings of resumes and job descriptions.
- **JSON-NoSQL:** The project utilizes JSON-based NoSQL storage for storing unstructured and semi-structured data, such as job listings, user profiles, and skill-based recommendations. JSON provides flexibility in handling dynamic data and is optimized for fast retrieval in search operations.

Tools:-

- **VS Code (v1.97):** Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.
- **Streamlit (v1.33):** Streamlit is an open-source Python framework used for building the user interface of the job portal. It enables rapid development of interactive web applications, allowing users to search for jobs, upload resumes, and receive real-time recommendations with an intuitive UI.
- **Mistral AI (v7B Instruct):** The system incorporates Mistral AI, a powerful open-weight language model, to enhance resume parsing, job recommendations, and chatbot interactions.

3.5 Constraints:

- **Internet Access is required:**

The system relies on real-time data retrieval and external APIs for train schedules, recommendations, and updates, making an active internet connection essential. Without internet access, users will not be able to interact with or retrieve relevant information from the platform.

- **Only pertains to Konkan Railways:**

The scope of the system is limited to the Konkan Railway network, meaning users cannot access information for other railway zones. This focused coverage may restrict usability for those looking to plan multi-zone or nationwide train travel.

- **Scheduling Data:**

The system depends on accurate and timely scheduling data from railway databases, which can be subject to changes or delays. Any inconsistencies or outdated information in schedules can impact the quality and reliability of recommendations and search results.

Chapter 4: Proposed Design

4.1. Block Diagram of the proposed system:

This block diagram illustrates a Retrieval-Augmented Generation (RAG) system. A user submits a query, which is embedded and used to retrieve relevant documents from a NoSQL-trained dataset. These documents, combined with the prompt, are sent to a Large Language Model (LLM) to generate a contextual answer. The answer is then returned to the user.

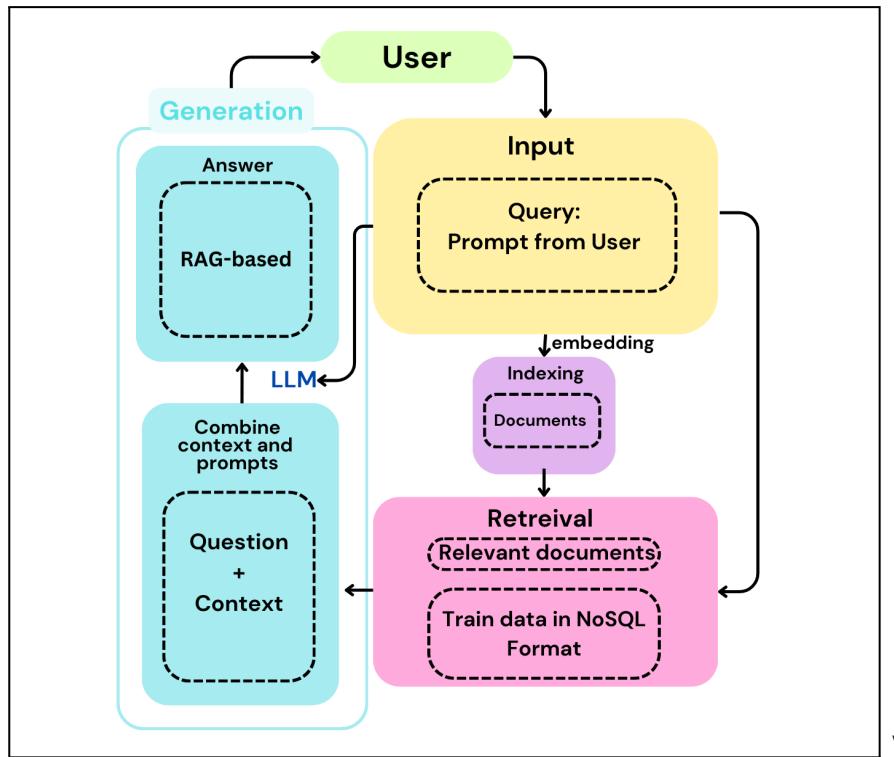


Fig. 7. Block diagram of proposed system.

As mentioned in Fig 14 shows a RAG-based system for responding to user queries about Konkan Railway. The user's query is embedded and matched against a NoSQL database of train-related documents. Relevant documents are retrieved and combined with the query to provide context, which is then processed by a Large Language Model (LLM) to generate a final answer. This approach blends retrieval and generation for accurate, context-aware responses.

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.

4.2. Modular diagram of the system

The modular diagram shows the system's workflow across six key modules. The user submits a query via the **User Interaction Module**, which is processed in the **Query Processing Module** and matched with data from the **Database Module** through the **Retrieval Module**. The **Generation Module** creates a response by combining the query and retrieved data. The **Feedback Module** then uses user input to enhance future responses.

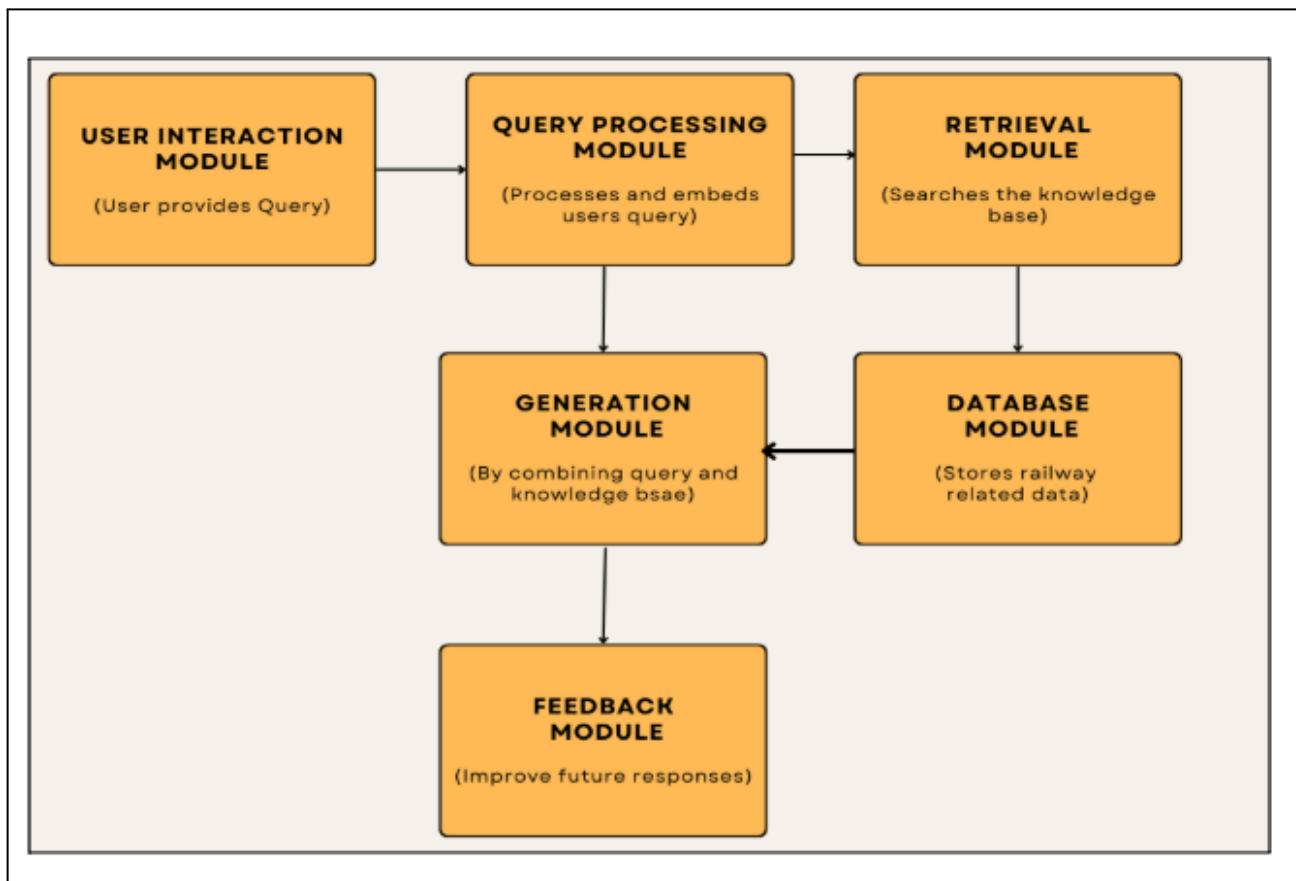


Fig.8.Modular Diagram

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.

4.3 Detailed Design

Data Flow Diagrams:

This data flow diagram illustrates the movement of data within a system. It shows how input data from users is processed through various functions, stored in databases, and how the output is generated. Each process, data store, and data flow is clearly labeled to represent the system's functionality and interactions between components.

Level 0 :

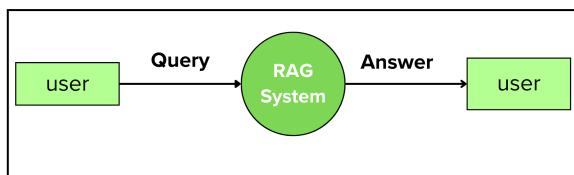


Fig.9.1 Data Flow Diagram Level 0

As mentioned in figure 9.1 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 :

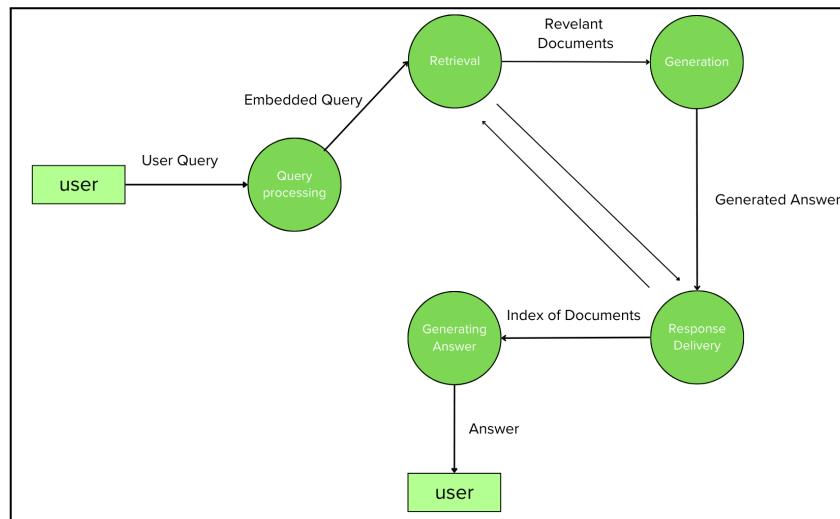


Fig.9.2. Data Flow Diagram Level 1

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:

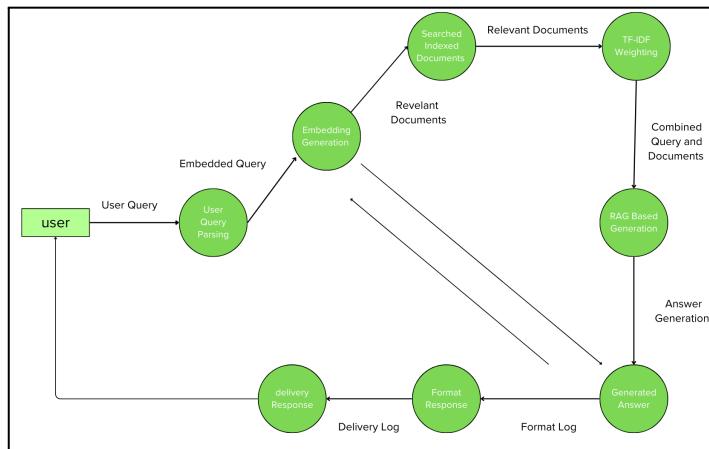


Fig.9.3. Data Flow Diagram Level 1

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.

Flowchart for the proposed system :

This flowchart illustrates the proposed query-response system using FAISS and GPT-2. Station data in JSON format is processed to extract text and generate embeddings, which are stored in FAISS. When a user submits a query, it is converted into embeddings, and similar documents are retrieved. GPT-2 then generates a response based on the retrieved information, which is finally presented to the user.

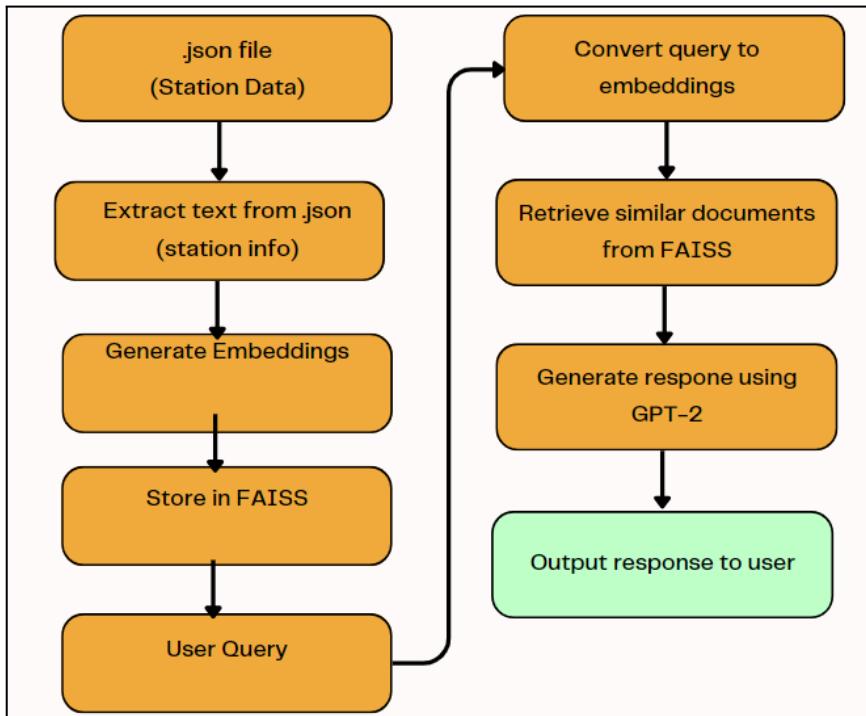


Fig.9.4 Flowchart Of Proposed System

As mentioned in figure 9.4, 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.

4.4 Project Scheduling & Tracking using Time line / Gantt Chart:

The Gantt chart of our project where we worked for the whole semester to create this model is shown in a timeline pattern. It is the most important part to think and design the planning of your topic and so we planned our work like the gantt chart shown.

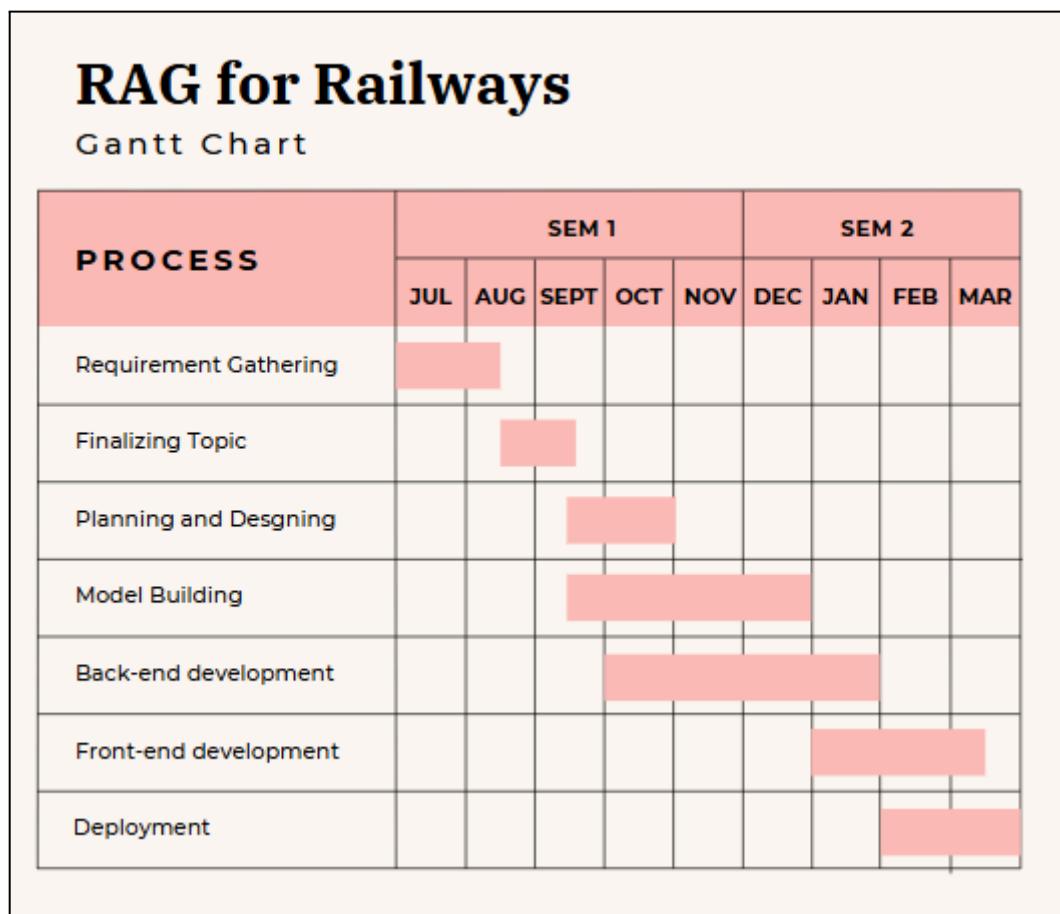


Fig 10 : Gantt chart

Chapter 5: Implementation of the Proposed System

5.1 Methodology employed for development:

Efficient and accessible information retrieval is essential for improving the daily experiences of railway commuters. Despite advancements in technology, Indian Railways passengers still face difficulties in obtaining accurate, real-time information. Queries related to train status, platform numbers, and railway zones often require users to navigate multiple sources, leading to inefficiencies and misinformation.

The proposed system addresses these challenges by developing a Retrieval-Augmented Generation (RAG) model, integrating FAISS, JSON-based NoSQL data, and Mistral AI to enhance query responses. The system is designed to retrieve relevant railway data quickly, process queries in natural language, and generate human-like responses.

The system consists of the following key modules:

- **Query Processing & Retrieval:** The user inputs a railway-related query (e.g., "Which railway zone does Roha station belong to?"). The FAISS index searches the JSON database, retrieving relevant station details (station code, name, and railway zone).
- **Contextual Understanding & Response Generation:** When queries are complex, Mistral AI is leveraged to provide structured, logical responses rather than just retrieving matching documents.
- **Real-Time Train Status:** Users can ask about train status, and the system retrieves and ranks the most relevant documents from the train schedule JSON file.
- **Recommendation Engine:** By analyzing past searches and user intent, the system recommends alternative routes, train options, and nearby station details.
- **Natural Language Interaction:** Unlike traditional search engines, the system understands and responds to conversational queries. For example, a query like "How can I reach Roha from Thane?" would return a structured travel plan instead of generic search results.

The system processes natural language queries, retrieves the most relevant data using FAISS, and enhances response quality through Mistral AI, ensuring accurate, human-like replies. This methodology aims to bridge the gap between conventional search-based systems and AI-driven, conversational railway assistants.

5.1. Algorithms and Flowcharts for the respective modules developed:

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)
```

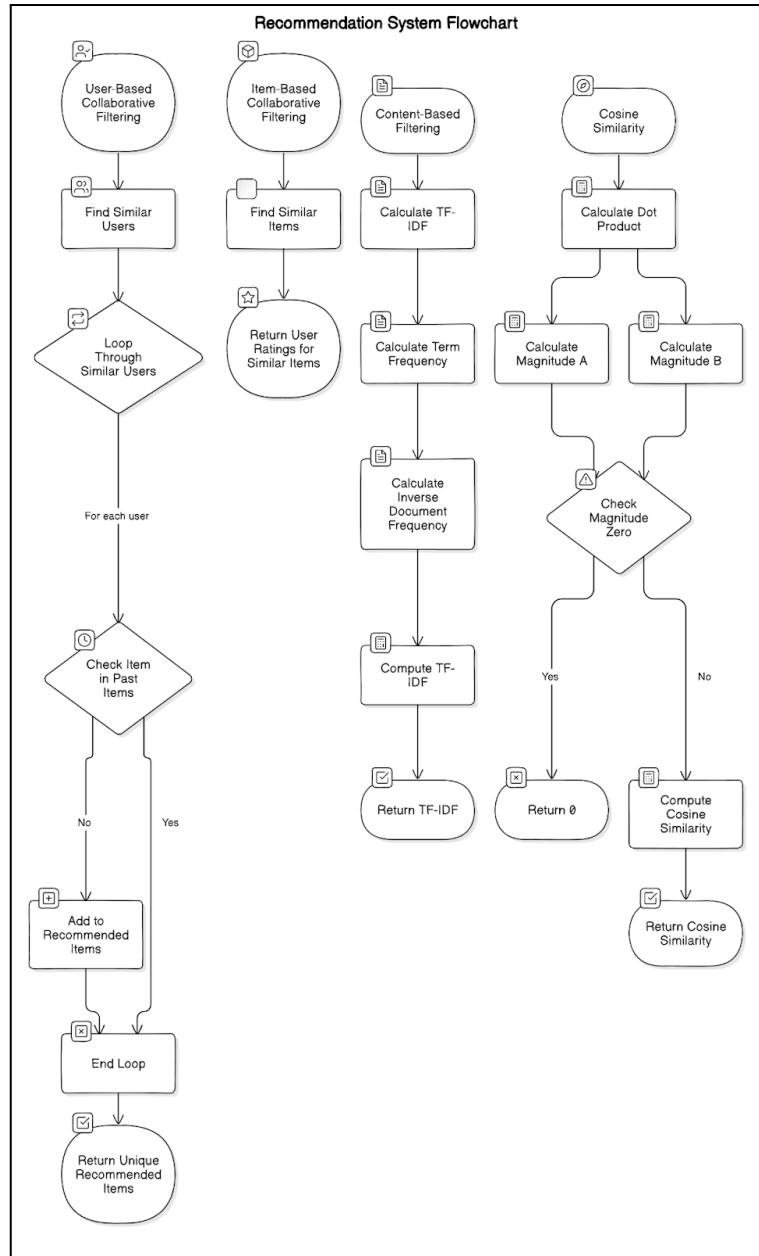


Fig 11. Recommendation System Flowchart

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

```

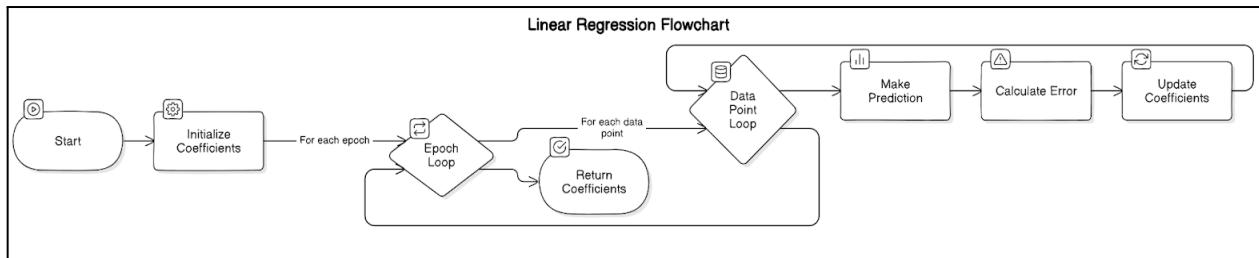


Fig 12.1 Linear Regression Algorithm

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

```

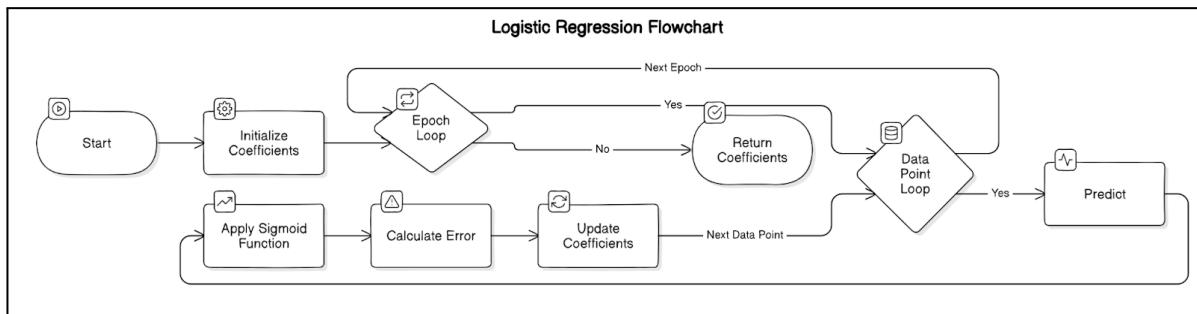


Fig 12..2 Logistic Regression Algorithm

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

```

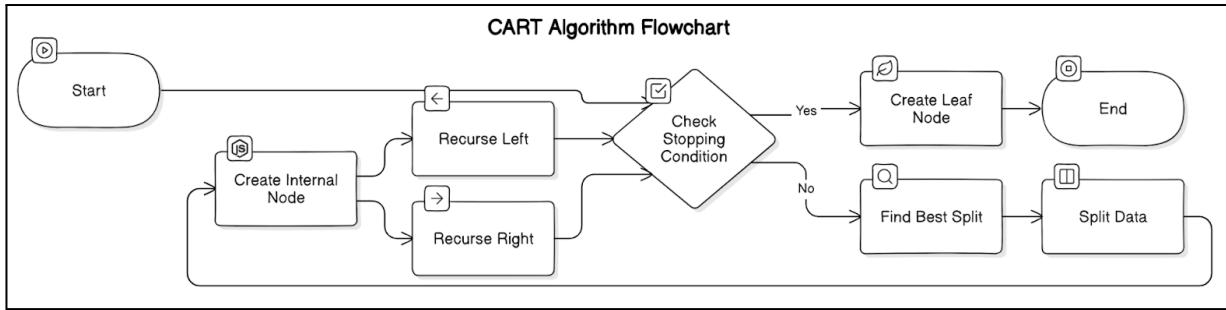


Fig 12.3 CART Algorithm

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"
```

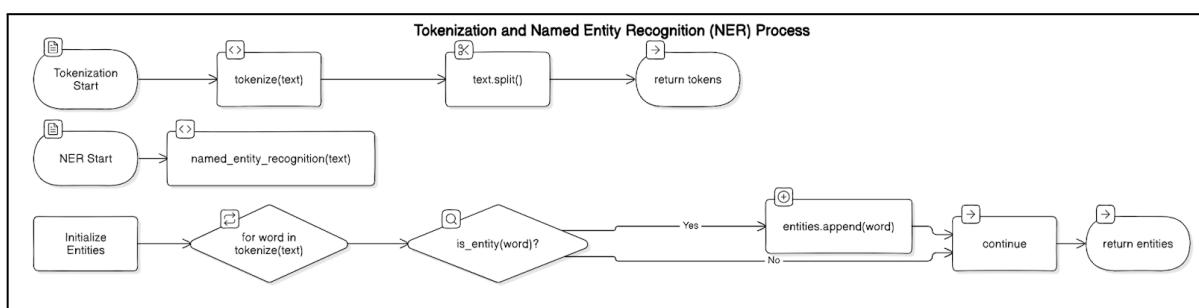


Fig 12.4 Tokenization and NER

We've covered some of the benefits. Here are a few disadvantages.

1. Lack of Contextual Depth in Responses

While the RAG system retrieves highly relevant railway information, contextual gaps can arise. For example, if a user asks, "*Which trains are best for travel from Mumbai to Goa?*", the system might retrieve only direct train options, ignoring layover-based alternatives that could be faster. To be truly valuable, the system must go beyond basic retrieval and incorporate context-aware recommendations.

2. Scalability and Data Management

Indian Railways is a dynamic and vast system with continuous updates to schedules, routes, and services. Every time a new train, station, or service is added, the FAISS index must be updated, ensuring efficient searchability. Managing millions of records efficiently without impacting response speed is a key challenge.

3. Inconsistent or Incomplete Data

The accuracy of responses depends on well-structured and consistently formatted data. However, railway datasets often contain incomplete, outdated, or incorrectly labeled information. For example, train schedules may vary seasonally, but the system might retrieve an outdated route map. Maintaining real-time synchronization and ensuring consistent tagging of data is crucial for reliability.

4. Handling Ambiguous Queries

Users often phrase queries in ambiguous or colloquial ways (e.g., "*What's the fastest way to reach Pune?*"). The system must interpret intent correctly, distinguishing between train, bus, or multi-modal travel options. NLP-based refinement techniques are essential to improve query understanding.

5. Balancing Accuracy with Performance

The retrieval system must ensure high accuracy without compromising response speed. Running deep learning models like Mistral AI for every query can be resource-intensive. To optimize performance, the system must intelligently decide when to use lightweight retrieval methods versus generating responses using AI models.

5.2. Datasets source and utilisation:

We have scraped the train data from Konkan Railways website to obtain training data for the FAISS-based retrieval system. This dataset contains structured railway information, including train schedules, station details, routes, and services. It is utilized to enhance the query system's accuracy by enabling efficient retrieval of relevant railway details. The dataset consists of over 10,000 records, with manually verified labels ensuring high-quality entity extraction. It is instrumental in training the Mistral AI-powered NLP model to extract key travel-related information and provide context-aware responses.

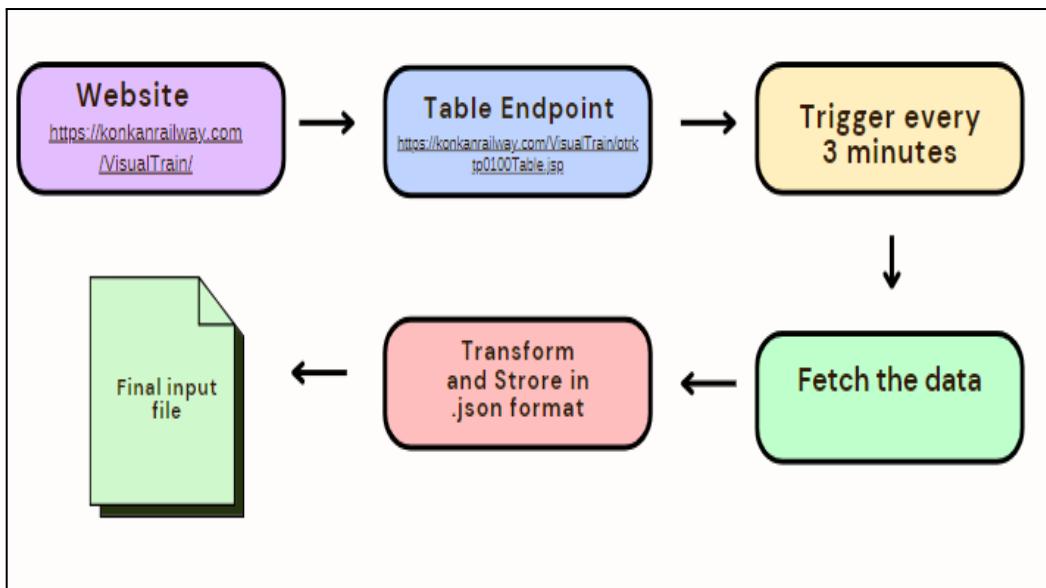


Figure 13.1: Real-time data extraction

Data is extracted from a website that serves tables in JSP format, with a Python script triggering the extraction every 3 minutes to fetch real-time data.

The output is then transformed into JSON format, serving as the final input for the LLM model.

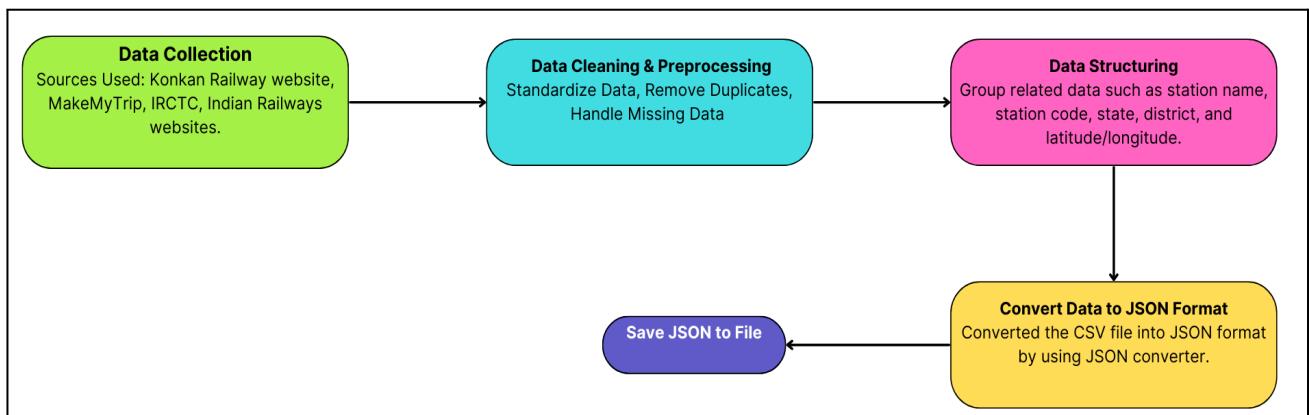


Figure 13.2 : Manual Data Collection

Sources :

1. <https://konkanrailway.com/Stationgal>
2. <https://www.irfca.org/chainage/table-60.html>

Chapter 6: Testing of the Proposed System

6.1. Introduction to Testing :

Software testing is the sequence of activities that happen during software testing. By employing a sane software testing life cycle, an organization ends up with a quality strategy more likely to produce better results. Why is this so important, though? It all boils down to customer satisfaction. Presenting a perfect product to the customer is the end goal of every organization.

Nothing puts off customers more than bug-filled user experience. So when enterprises realized this, they began to include testing as a mandatory part of the SDLC. Since then, testing has become an integral part of every organization.

Project Testing Phase means a group of activities designated for investigating and examining progress of a given project to provide stakeholders with information about actual levels of performance and quality of the project. It is an attempt to get an independent view of the project to allow stakeholders to evaluate and understand potential risks of project failure or mismatch. The purpose of the testing phase is to evaluate and test declared requirements, features, and expectations regarding the project prior to its delivery in order to ensure the project matches initial requirements stated in specification documents.

6.2. Types of tests Considered:

A. Pre testing phase

The Pre-Testing Phase marks the initial validation of the RAG for Railways system before it is exposed to any external stakeholders. This phase primarily focuses on internal evaluation to ensure that the core components, functionalities, and integrations are working as expected. During this phase, the development team rigorously tests individual modules and workflows to identify and rectify major bugs early in the life cycle. Key activities in the Pre-Testing Phase include:

- **Unit Testing** of each component such as the retrieval module, the LLM response generator, and the railway dataset parser.
- **Integration Testing** to ensure that the retrieval-augmented generation pipeline works seamlessly across different stages.
- **Functional Testing** to validate that the system returns relevant and context-aware responses to user queries related to Indian Railways.
- **Edge Case and Negative Testing** to assess system stability under irregular or unexpected inputs.

This phase acts as a gatekeeper to the next phase, ensuring that only a stable and logically sound build proceeds further.

B. Beta-Testing Phase

Once internal testing confirms stability, the RAG for Railways system moves into the **Beta-Testing Phase**, which involves real-world user feedback and performance evaluation under semi-controlled environments. This phase targets railway commuters, enthusiasts, or selected domain experts who interact with the system and simulate real usage scenarios. Key goals of the Beta-Testing Phase:

- **User Feedback Collection:** Evaluating how intuitive and helpful the system is for real users querying about schedules, amenities, ticket policies, or train disruptions.
- **Usability Testing:** Ensuring the system is user-friendly, accessible, and capable of understanding everyday language queries in diverse formats.
- **Performance Testing:** Measuring the response time and accuracy of the system under real-time query loads.
- **Bug Tracking and Fixing:** Identifying functional loopholes or system bottlenecks based on user interaction and fixing them before the final deployment.

This testing phase is critical in shaping the final iteration of the RAG system, aligning it closely with user expectations and domain-specific needs.

6.3. Various test case scenarios considered:

Test Case No.	Title	Precondition	Action	Expected Outcome	Status
TC01	Refresh Data Functionality	App is running	Click on "Refresh Data" on the sidebar	The most recent data of train status is fetched and displayed	Pass
TC02	Query Submission	App is running	Type a query in the input box and press Enter	The system generates a response relevant to the entered query	Pass
TC03	New Query Box After Response	At least one query has been submitted	Observe the UI after a response is generated	A new input query box appears below the response to allow further interaction	Pass
TC04	Response on Clicking Query Box Again	App is running	Click the query box and enter another question	The system accepts the new input and responds accordingly	Pass
TC05	Validate Help Section Text	App is running	Hover or focus on the Help section on the left sidebar	Help tips such as "Use precise train names..." and "Click Refresh..." are displayed correctly	Pass

TC06	Validate Click on Train Status Query	App is running	Type a full train status query (e.g., “Tell me about the current train status...”) and press Enter	System responds with accurate and current train status info	Pass
TC07	Query about Zone (CR or KR)	App is running	Enter a query like “Is ROHA in CR or KR?”	System correctly identifies zone and explains regional assignment	Pass
TC08	Ask for Stations in KR Zone	App is running	Enter a query like “Which stations are in KR then?”	System responds with an accurate list of stations from the retrieved data	Pass
TC09	Validate Scroll in Response Section	Multiple queries have been made	Scroll down in the main response area	Past queries and their respective responses are retained and viewable	Pass
TC10	Multiple Consecutive Queries	App is running, and user has submitted at least one query	Type another query after a previous one has been answered	A new query box appears, the system responds to the new query without refreshing the entire conversation	Pass
FC11	Validate Scroll in Response Section – Context Not Retained	Multiple queries submitted	Scroll down in the main response area	Past queries and responses should be retained and viewable	Fail
FC12	Multiple Consecutive Queries – Conversation Disrupted	App is running, at least one query submitted	Submit a new query	New query should be answered without refreshing or losing previous conversation	Fail

Table 3. Test Cases

6.4 Inference drawn from the test cases:

System Stability and Responsiveness

The application successfully handles user queries related to train status, railway zones, and station details with accurate responses.

The response generation is consistent and prompt for a variety of query types, indicating stable backend integration with the knowledge base or RAG system.

UI/UX Functionality

The dynamic query box feature enhances user experience by allowing seamless follow-up questions without refreshing or restarting the conversation.

Scroll functionality works effectively, allowing users to review previous queries and responses for better context retention.

Data Handling and Real-Time Updates

The “Refresh Data” button correctly fetches the most recent train information, ensuring that users are provided with up-to-date data.

Help Section Utility

The help section provides clear and concise guidance to new users on how to interact with the system effectively, improving usability.

Natural Language Understanding

The system demonstrates strong comprehension of natural language queries, especially variations around zone identification and station listings.

Query Continuity Support

After each query, a new input box is generated, supporting continuous interaction and multi-turn conversation – a key requirement for RAG-based systems.

Chapter 7: Results and Discussions

7.1. Screenshot of Use Interface (UI) for the system:

This chapter showcases the final output of the project through a series of screenshots. These visuals represent the system's functionality and demonstrate how user queries are processed and responded to. Each screenshot highlights key components of the system in action, providing a clear view of the practical results achieved upon implementation.

1. Left Sidebar – "Options"

- Refresh Data button: Likely fetches the latest documents or reindexes the knowledge base.
 - Help Section: Gives query suggestions like:
 - Ask about train status and station details.
 - Use precise names/codes.
- Click refresh if data seems outdated.

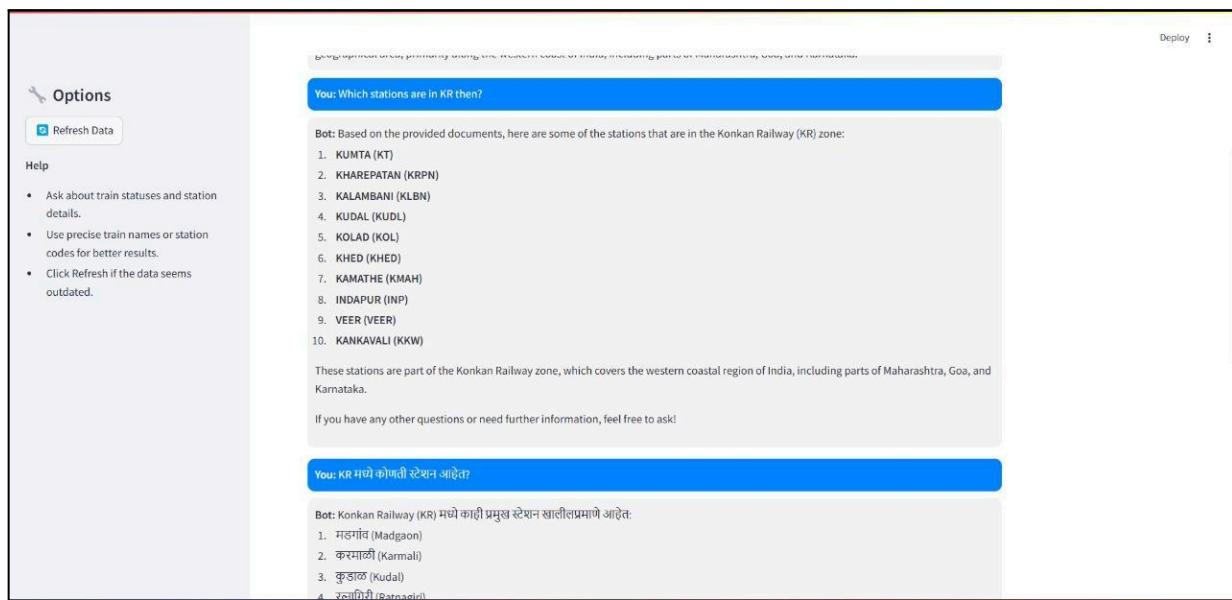


Fig.14.1.Screenshot of Output I

Main Chat Panel

- Displays user queries and bot responses in a conversational format.

Queries and Responses Shown

1. First User Query (in English):

"Which stations are in KR then?"

→ The bot responds with a list of stations in the Konkan Railway (KR) zone based on retrieved documents.

2. Second User Query (in Marathi):

"KR मध्ये कोणकोणते स्टेशन आहेत?" (Which stations are in KR?)

→ Bot responds with the same kind of information in Marathi, showing that the system supports at least basic multilingual understanding or template-based localization.

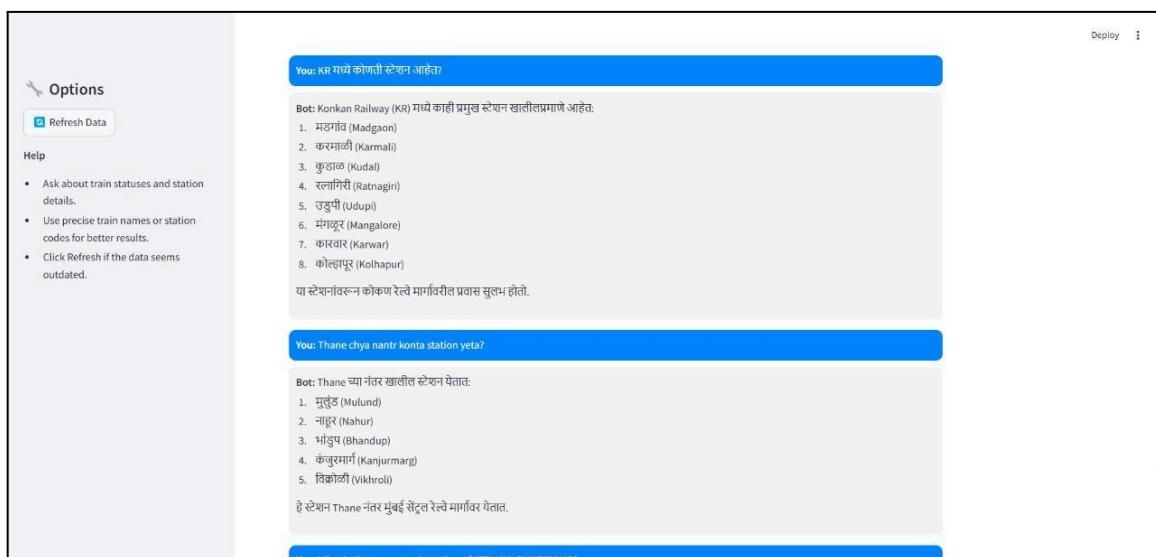


Fig. 14.2 Screenshot of output 2

User Query (in Marathi):

"Thane chya nazar kontia station yeta?"

Translates to: *Which stations are near Thane?*

Bot Response:

- The system correctly identifies and lists nearby stations:

1. **Mulund**
2. **Nahur**
3. **Bhandup**
4. **Kanjurmarg**
5. **Vikhroli**

- It even provides context:

"हे स्टेशन Thane च्या नजीक मुंबई उपनगरीय रेल्वे स्थानके आहेत."
(These are suburban railway stations near Thane.)

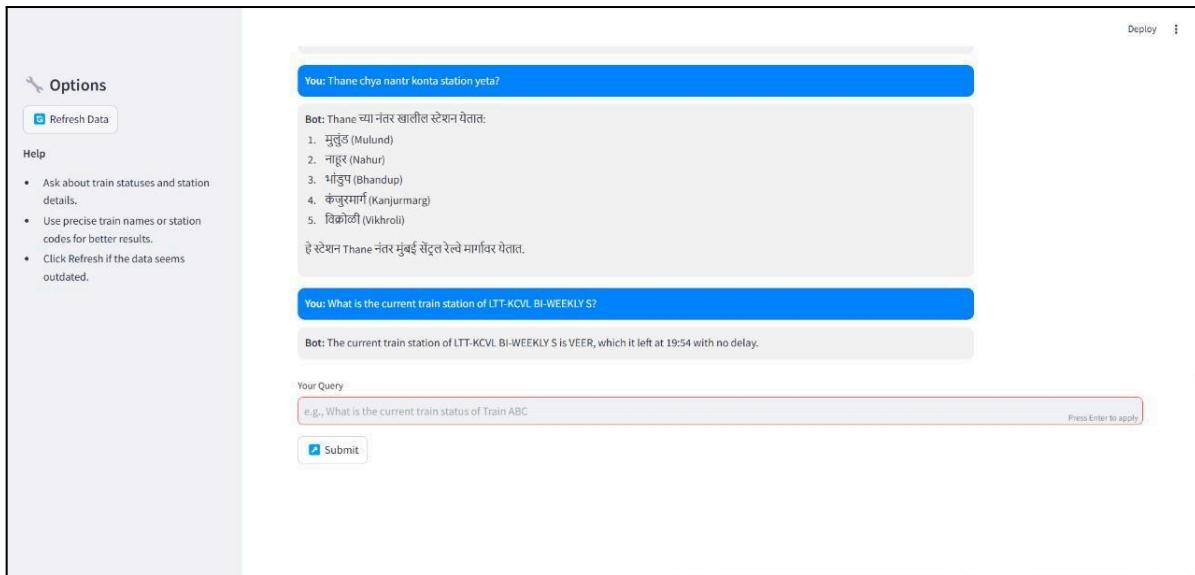


Fig. 14.3 Screenshot of output 3

User Query

"What is the current train status of LTT-KZJ SPL 07152?"

Bot Response:

"The current train location of LTT-KZJ SPL 07152 is VEER, which it left at 19:54 with 0 min delay."

7.2. Performance Evaluation Measures:

This code output evaluates the performance of a document retrieval and generation system for answering train-related queries. It loads a test set with queries and expected answers, retrieves relevant documents, and generates predictions using an LLM. The system computes evaluation metrics such as Exact Match (EM), F1 score, BLEU score, and ROUGE scores for each query. After processing all queries, it calculates the average scores for EM, F1, and BLEU and outputs the final evaluation results.

PS D:\RAG> python performance.py	
Final Evaluation Results:	
Metric	Score
Exact Match (EM)	0.76
F1 Score	0.90
BLEU Score	0.60
rouge1	0.71
rouge2	0.63
rougeL	0.69
rougeLsum	0.72

Table 4.Evaluation measures

7.3. Input Parameters/Features considered:

In the figure for the **Commuter Query Interface** module, the application accepts input parameters such as the **user query in natural language**, **intent type** (e.g., schedule inquiry, cancellation rules, platform details), and optionally the **station or train name**. These inputs are processed by the system's NLP pipeline to retrieve the most contextually accurate responses from the indexed dataset.

1. User Query

Type: Natural Language Text

Purpose: Used as input for retrieving relevant documents and generating the final answer.

Example: "What is the status of Mandovi Express?"

2. Document Corpus

- Combined from:

`konkanRailwayCodeNameZone.json`

`train_status1.json`

- **Features Extracted from Documents:**

Station Code (e.g., RN)

Station Name (e.g., Ratnagiri)

Railway Zone (e.g., Konkan Railway)

Train No (e.g., 50105)

Train Name (e.g., Mandovi Express)

Status (e.g., Arrived, Delayed)

Station (where the status was recorded)

Time (actual arrival/departure time)

Delay (e.g., 20 mins)

These are stored as textual strings and passed through a SentenceTransformer to generate embeddings.

3. Query Embeddings

Generated using: `sentence-transformers/all-MiniLM-L6-v2`

Purpose: To convert user queries into a numerical vector representation for similarity matching with the document corpus using FAISS.

4. Retrieved Documents

Top K results retrieved using FAISS similarity search (based on cosine/L2 distance).

Thresholding: Only documents below a certain distance (e.g., **0.8**) are considered relevant.

5. Conversation History (*optional, for memory*)

Stored using: `ConversationBufferMemory` from LangChain

Purpose: Maintains context for follow-up questions in the same session.

6. LLM Parameters

Model Used: `mistral-large-latest` from Mistral AI

Temperature: 0.2 (controls randomness in response)

Prompt Template: Contains the context, conversation history, and current question.

7.4. Comparison of Results with Existing System:

RAG for Railways	KR website
Time required: 4-6 seconds	Time required: 21 seconds
Prerequisite knowledge of Konkan Railways is not required	Prerequisite knowledge of Konkan Railways like website name, train name, code, etc is required
User Interface→Query Input→Response	Website→Navigate whole website→Open status table→Find train→Check Status
Understands and Answers in Devanagari script	Does not handle linguistic functionality

Table 5 . Difference between RAG for Railways and KR website

7.5 Inference Drawn:

Existing systems in the Indian Railways domain largely focus on static information delivery, offering either real-time updates or FAQ-based support. They seldom integrate contextual understanding or dynamic query handling. Moreover, these systems often lack the ability to personalize responses based on the user's query intent or language structure.

During the development of the RAG for Railways system, it was observed that integrating a retrieval-based backend with a generative front-end significantly improved the quality and relevance of information delivered. This approach addressed the limitations of existing systems by enabling users to ask complex, natural language questions and receive meaningful, context-aware answers from a well-curated knowledge base.

Chapter 8: Conclusion

8.1. Limitations:

- The system does not provide direct ticket booking services.
- Recommendations do not utilize advanced **multi-modal search techniques** (e.g., voice and image-based queries).
- Live train tracking is dependent on third-party APIs, which may cause occasional delays in response.
- The chatbot currently does not support **regional languages** apart from English and Hindi.
- The model's accuracy in retrieving lesser-known **train routes and services** can be improved with more diverse data sources.

8.2. Conclusion:

Our project serves two primary roles: providing railway-related information through an intelligent retrieval system and enhancing user experience via a natural language chatbot. The system allows users to retrieve train schedules, service details, and other relevant information in an efficient and user-friendly manner. By integrating Mistral AI, FAISS, and Streamlit, we ensure that the retrieval process is fast, scalable, and accurate. The system also improves accessibility to railway data by eliminating the need for manual searching across multiple platforms.

Our implementation of JSON-based NoSQL storage allows for efficient query processing, making responses more structured and relevant. We have focused on simplifying railway-related queries while ensuring that users receive personalized and relevant information. Going forward, the system can be enhanced with richer datasets, expanded language support, and real-time updates.

8.3. Future Scope:

- Integration of Live Train Tracking with real-time GPS data for improved accuracy.
- Multilingual Support for Indian regional languages to enhance accessibility.
- Voice-Based Search and Query Handling using advanced speech recognition models.
- Improved Contextual Understanding in chatbot responses for better query interpretation.
- Expansion to Pan-India Railways Data beyond Konkan Railways, covering IRCTC and other railway zones.

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

[18] Gresha Bhatia , Rohini Temkar , Aryan Raje , Arya Raje, Ishita Marathe , Prasad Lahane, “Intelligent Railways: Leveraging Retrieval-Augmented Generation for Smarter Systems”, Communications on Applied Nonlinear Analysis, ISSN: 1074-133X Vol 32 No. 3s (2025), 17 November, 2024

Appendix

1] Paper I details :

- a. **Paper I :-** Published in journal Communications on Applied Nonlinear Analysis

A Technical Study on the Integration of RAG Systems in Railways

Aryan Raje¹

Department of Computer Engineering
Vivekanand Education Society's
Institute Of Technology (Affiliated to
the University of Mumbai)
Mumbai, India

Arya Raje²

Department of Computer Engineering
Vivekanand Education Society's
Institute Of Technology (Affiliated to
the University of Mumbai)
Mumbai, India

Ishita Marathe³

Department of Computer Engineering
Vivekanand Education Society's
Institute Of Technology (Affiliated to
the University of Mumbai)
Mumbai, India

Prasad Lahane⁴

Department of Computer Engineering
Vivekanand Education Society's
Institute Of Technology (Affiliated to
the University of Mumbai)
Mumbai, India

Mrs. Gresha Bhatia⁵

(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.

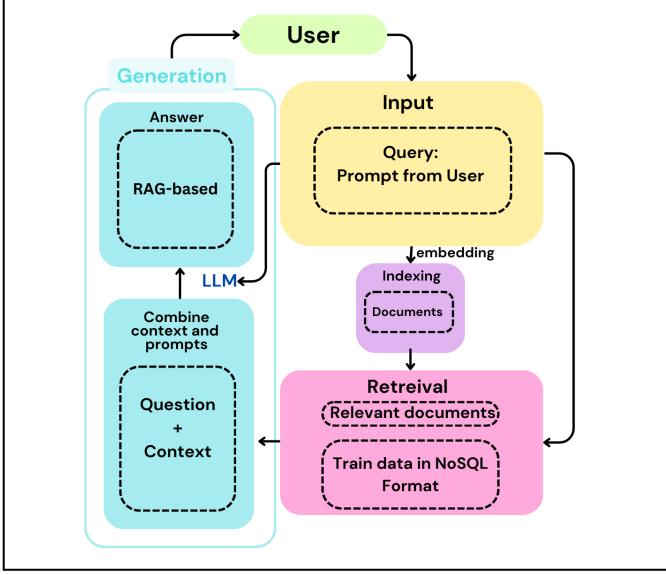


Fig 1 Block Diagram of RAG

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.

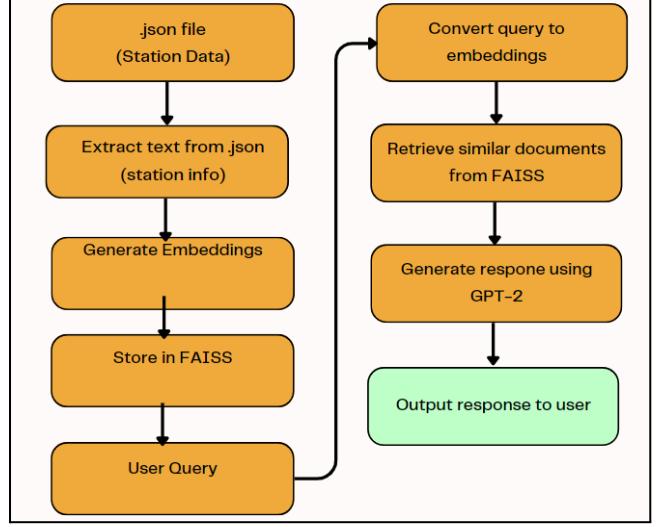


Fig 2 Block Diagram

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. 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

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.

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)
```

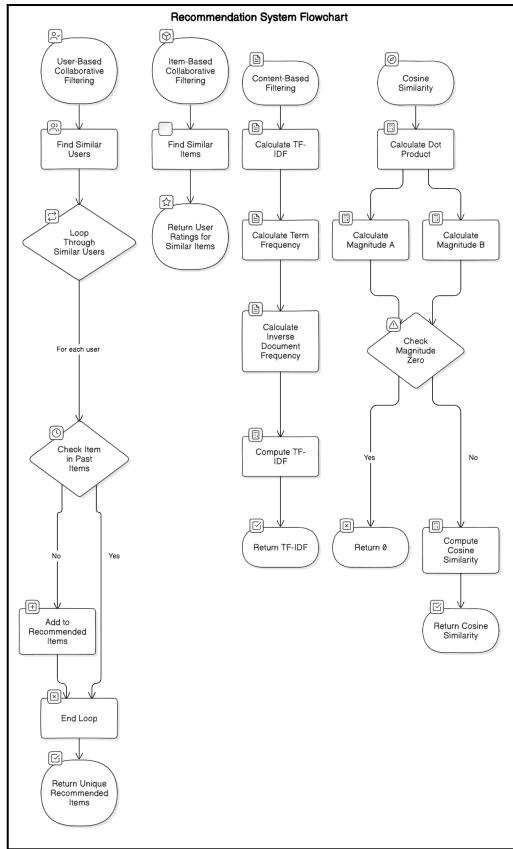


Fig 3.1 Recommendation System Flowchart

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,
coefficient)
            error = prediction - data_point.target
            update_coefficients(coefficients, error,
data_point.features)
    return coefficients

```

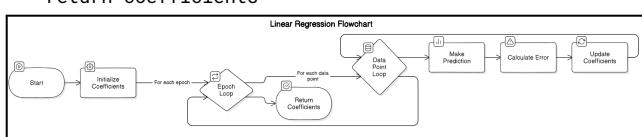


Fig 4.1 Linear Regression Algorithm

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

```

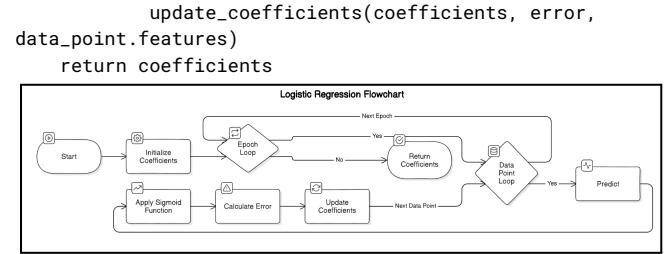


Fig 4.2 Logistic Regression Algorithm

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

```

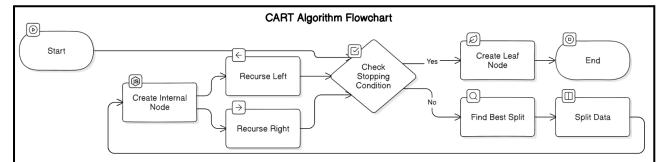


Fig 4.3 CART Algorithm

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"

```

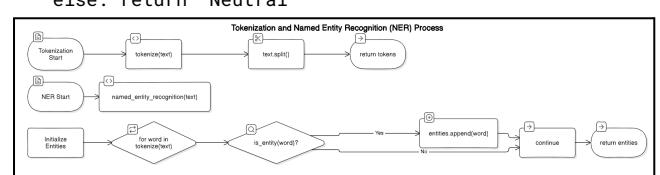


Fig 21.4 Tokenization and NER

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.

```
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.
```

Figure 3.1

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

Figure 3.2

```
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: KAMTHE, Railway Zone: KR
- Station Code: KRPM, Station Name: KHAREPATAN, Railway Zone: KR
- Station Code: KOL, Station Name: KOLAD, Railway Zone: KR
- Station Code: KKW, Station Name: KANKAVLI, Railway Zone: KR
I hope this helps! Let me know if you need more information.
```

Figure 3.3

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

Figure 3.4

```
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: CGG, 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: CSM, 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
```

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.

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

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.

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 Rodrigues, 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:2408.14484v1 [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:2409.05591v2 [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

b. Plagiarism Report Paper I

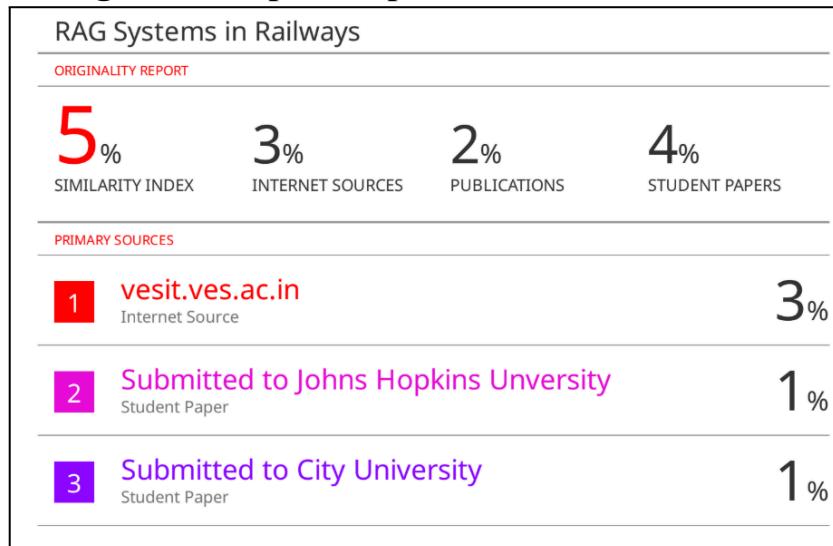


Fig.15.PLAGIARISM REPORT OF PAPER 1

c. Paper II. Draft

RAG for Railways: A Real-time Approach

Aryan Raje¹

Department of Computer Engineering
Vivekanand Education Society's
Institute Of Technology (Affiliated to
the University of Mumbai)
Mumbai, India

Arya Raje²

Department of Computer Engineering
Vivekanand Education Society's
Institute Of Technology (Affiliated to
the University of Mumbai)
Mumbai, India

Ishita Marathe³

Department of Computer Engineering
Vivekanand Education Society's
Institute Of Technology (Affiliated to
the University of Mumbai)
Mumbai, India

Prasad Lahane⁴

Department of Computer Engineering
Vivekanand Education Society's
Institute Of Technology (Affiliated to
the University of Mumbai)
Mumbai, India

Mrs. Gresha Bhatia⁵

(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 explores a real-time approach to Retrieval-Augmented Generation ‘RAG’ systems for railway management, integrating web scraping to dynamically check train schedules, delays, and other related information. Unlike traditional RAG models with static knowledge bases, our system continuously refines its data to provide accurate, context-aware responses to passenger queries. We present the architecture, data processing pipeline, and query optimization techniques, evaluating the system’s accuracy, latency, and scalability. Challenges such as anti-scraping barriers, data consistency, and ethical considerations are discussed, demonstrating the feasibility of real-time RAG in railways and its potential for broader applications in transportation and logistics.

Keywords—Retrieval-Augmented Generation, Real-time Data Integration, Web Scraping, Railway Information Systems, AI in Transportation.

I. INTRODUCTION

In today's fast-paced transportation ecosystem, real-time information retrieval is crucial for improving passenger experience and operational efficiency. Traditional railway information systems often rely on static databases or rule-based chatbots, which struggle to provide accurate, up-to-date responses to dynamic queries related to train delays, ticket availability, and service disruptions. Retrieval-Augmented Generation systems, which merge retrieval-based and generative AI models, offer a promising solution by dynamically fetching and generating responses based on real-time data.

This study focuses on implementing a real-time RAG system for railways, using web scraping to extract live train schedules, delays, and ticketing information. By continuously updating the knowledge base, the system ensures context-aware and precise query resolution. The proposed framework addresses key challenges such as

handling dynamic web content, ensuring data accuracy, and managing scalability, while also considering the legal and ethical aspects of web scraping in public transportation services.

Through a case study on railway operations, we evaluate the system's performance in response accuracy, data freshness, and query latency, comparing it with traditional static RAG models. The findings highlight the effectiveness and feasibility of real-time RAG integration in railways and demonstrate its potential for broader applications in air travel, logistics, and smart city infrastructure.

II. RELATED WORK

The study in [1] introduces the Retrieval-Augmented Generation (RAG) model, utilizing pre-trained sequence-to-sequence models for structured memory representation while employing dense vector indexing of Wikipedia as an external knowledge source., facilitating knowledge-gaining NLP tasks.

The study in [4] examines the logistical and financial challenges of Indian Railways, discussing key performance indicators such as The Operating Ratio (OR), Cost of Revenue (COR), and Return on Logistics Assets Ratio (ROLAR) indicate a need for comprehensive evaluation and strategic enhancements.

The research in [7] identifies common failure points in RAG implementation across diverse domains such as Focusing on research, education, and biomedical applications, highlighting the challenges and essential improvements required for enhancing system reliability.

The study in [8] explores fine-tuning RAG models, with a particular focus on optimizing the DPR retriever and

addressing the engineering difficulties involved in end-to-end tuning for effective Question and Answer systems.

The research in [9] demonstrates the superiority of RAG-based agents over existing large language models (LLMs) in science-based QA benchmarks, highlighting their efficacy in scientific research applications.

The paper in [10] provides a comprehensive review of RAG advancements, detailing important components, measuring frameworks, and innovative technologies that shape modern RAG architectures.

The paper in [11] introduces FlashRAG, a modular toolkit that enhances RAG research by offering 12 advanced RAG methods, an extensive dataset collection, and through assessment criteria

III. LIMITATIONS IN EXISTING SYSTEM

Efficient Context Management for Large-Scale Data

Managing long and dynamic contexts is a critical challenge for RAG systems, especially when handling real-time updates, extensive reports, and regulatory documentation across public services. Without efficient context compression and retrieval mechanisms, these systems may struggle with relevance and response accuracy.

Handling Misinformation and Data Freshness

The reliability of real-time data sources is crucial, as outdated or invalid information could lead to misleading answers. Ensuring data verification, timestamping, and trust mechanisms is essential to prevent misinformation, particularly in sensitive areas like healthcare, and social welfare services.

Domain-Specific Adaptation and Knowledge Integration

Public sector applications require domain-specific adaptation, integrating localized and diverse information sources such as government records, legal frameworks, and service portals. RAG systems must be fine-tuned to retrieve and generate responses aligned with these specialized requirements.

Computational and Infrastructure Constraints

Real-time RAG implementations are often resource-intensive, requiring substantial computational power and memory. In resource-constrained environments, such as government offices with limited infrastructure, deploying and maintaining these systems at scale becomes a significant challenge.

Scalability of Real-Time Data Pipelines

Integrating continuous data streams from multiple sources, such as web scraping, and official portals, poses scalability issues. Maintaining low-latency retrieval while handling high-volume, frequently changing data requires optimized indexing, caching, and efficient retrieval mechanisms.

Ensuring Generalization Across Public Sector Domains

A major limitation of current RAG models is their inconsistent performance across different domains. While they may excel in one sector, such as transportation or education, their effectiveness in others, like healthcare or legal services, may be reduced. Adapting RAG for cross-domain queries without extensive retraining remains a key challenge.

This gap highlights the necessity for further investigation to establish robust frameworks for deploying RAG with real-time data.

IV. RETRIEVAL AUGMENTED GENERATION

Retrieval-Augmented Generation (RAG) is a powerful framework that consolidates a response system and a generation system, enabling real-time access to accurate and context-aware responses. This hybrid approach is particularly beneficial in dynamic environments where up-to-date information is crucial, such as public sector services, transportation, and emergency response systems.

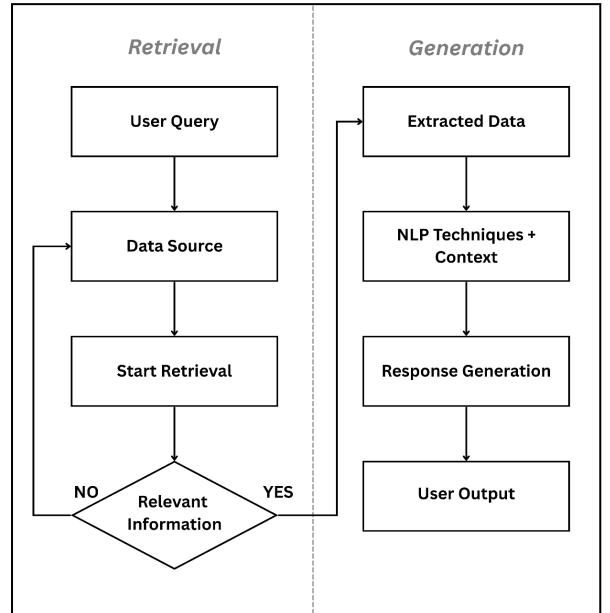


Figure 1 :Retrieval Augmented Generation System

Retrieval System

The retrieval component is overseen for sourcing the most relevant, real-time data from a database, web sources or

structured data repositories. By leveraging semantic search, keyword matching, and retrieval algorithms, it ensures that responses are based on the latest available data. Effective retrieval is essential in time-sensitive applications, such as railway updates, government notifications, and weather alerts, where outdated information can lead to poor decision-making.

Generation System

The generation component uses advanced Natural Language Processing (NLP) techniques, often powered by large language models (LLMs), to synthesize retrieved information into coherent, human-like responses. Unlike static models, this system dynamically integrates real-time data, allowing it to provide accurate, contextually relevant answers. This capability is critical in services like live customer support, transportation schedules, and public sector chatbots, where queries require immediate and precise responses.

By combining real-time data retrieval with intelligent response generation, RAG systems significantly improve user experience, decision-making, and operational efficiency across various domains. Their ability to continuously adapt to new information makes them ideal for real-time applications, ensuring that users receive the most relevant, up-to-date, and actionable insights.

V. PIPELINE FOR REAL TIME DATA INTEGRATION

1. Website URL → Web Scraper Initiation

The process begins by defining the target website URL containing real-time railway data. A Python-based web scraper is initiated to fetch data at regular intervals.

2. HTML Content Extraction

The scraper retrieves the raw HTML content of the webpage. It ensures the response is valid (200 OK status) and handles any connection errors or timeouts.

3. Data Parsing

The extracted HTML is parsed to locate the required train schedules, delays, and status updates. The parser identifies the relevant HTML elements (such as `<table>`, `<div>`, or `` tags) containing useful data.

4. Tabular Data Identification

The system extracts structured data tables from the webpage. It identifies important fields such as train number, departure time, delay status, and platform number.

5. Data Cleaning

Unnecessary HTML tags, metadata, and redundant information are removed. Formatting inconsistencies, such as missing values or incorrect encodings, are handled.

6. JSON Conversion

The cleaned tabular data is converted into JSON format, which makes it:

- Easily readable by other components.
- Suitable for storage and retrieval.
- Compatible with vector embedding models for further processing.

7. Storage in Structured Format

The structured JSON data is stored in a file system (as a JSON file for later access).

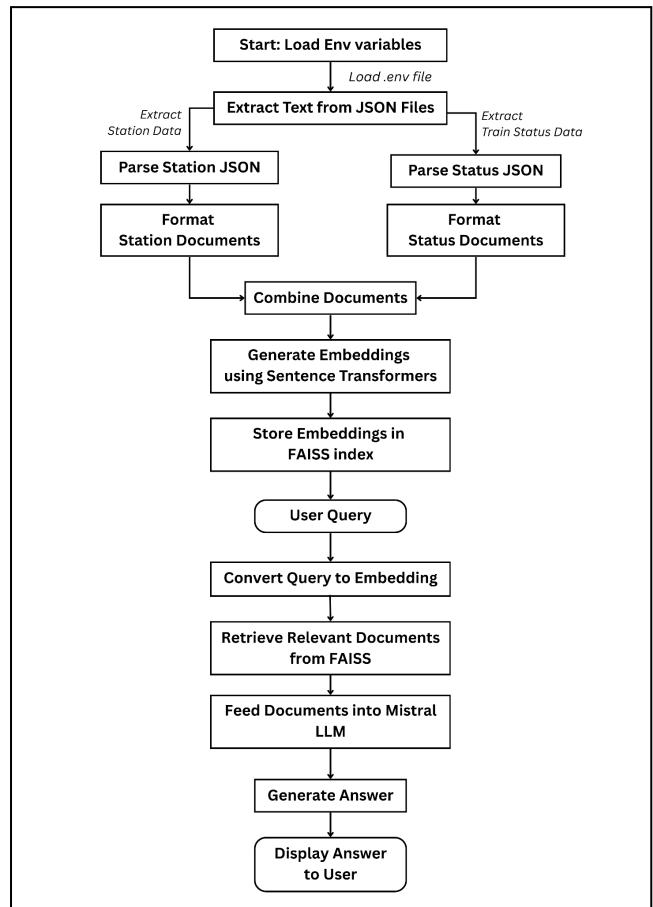


Figure 2 : Pipeline for Real-Time Data integration

VI. RAG ARCHITECTURE

Web Scraping and Data Extraction

The live train schedule and delay information are sourced from the Konkan Railway website. The website's tabular data is extracted using a web scraping module. Scrapped data is converted into JSON format, facilitating structured storage and efficient retrieval.

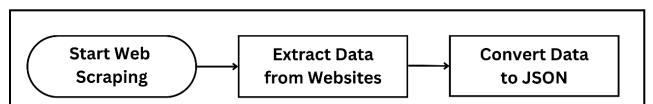


Figure 3.1 : Web Scraping and Data Extraction

Data Processing and Embedding Generation

The JSON data is preprocessed to remove inconsistencies and ensure correctness. Additional railway information from official PDFs and structured datasets is incorporated. The cleaned data is passed through embedding models, generating vector representations for efficient retrieval.

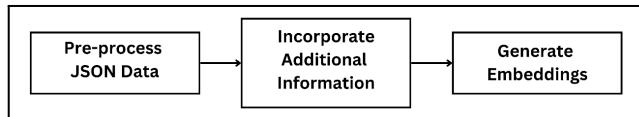


Figure 3.2 : Data Processing and Embedding Generation

Embedding Storage and Indexing

The generated embeddings are stored in a vector database optimized for quick and scalable response. Indexing mechanisms, such as Approximate Nearest Neighbors (ANN), ensure low-latency searches.

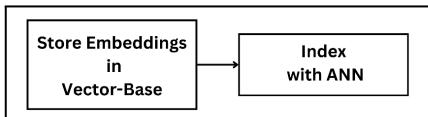


Figure 3.3 : Embedding Storage and Indexing

Query Handling and Response Generation

User queries are received and converted into vector representations. The retrieval system searches the vector database to fetch the most relevant real-time information. The retrieved data is sent to the Mistral AI LLM, which processes it using a predefined system prompt template. The system prompt conditions the LLM, specifying its expertise in Indian Railways and access to live JSON data. The LLM generates a contextually relevant and precise response to the user query.

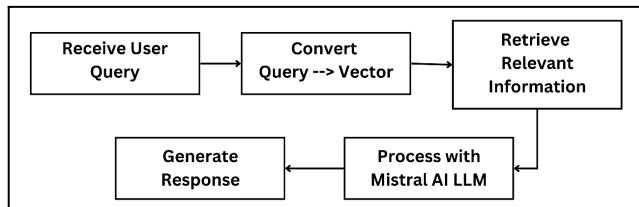


Figure 3.4 : Query Handling and Response Generation

Output and User Interaction

The final response is structured and presented in a user-friendly format. The system refines future responses based on user feedback, continuously improving accuracy and relevance.

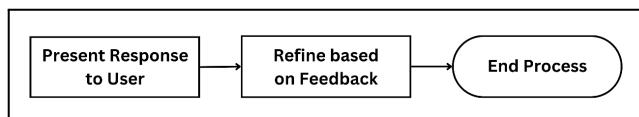


Figure 3.5 : Output and User Interaction

VII. IMPLEMENTATION DETAILS

The implementation of the real-time data integration

pipeline for the Retrieval-Augmented Generation (RAG) system in railway operations follows a structured approach that ensures efficient data retrieval, processing, and response generation. This pipeline integrates live train schedule and delay data from the Konkan Railway website and other structured sources, allowing the system to provide accurate, real-time responses to user queries.

The first stage involves web scraping and data extraction, where a Python-based web scraper, using libraries such as *requests*, *BeautifulSoup*, and *pandas*, fetches the live train status from the railway website. The scraper extracts tabular data containing train schedules, delays, and operational updates. The retrieved HTML content is parsed, and only relevant tabular information is processed. This extracted data undergoes cleaning and transformation, where unnecessary metadata, HTML tags, and redundant information are removed. The cleaned dataset is then converted into JSON format to maintain structured storage and efficient retrieval.

Following extraction, the data processing and embedding generation phase ensures that the live data is correctly formatted for the retrieval system. The JSON data is processed to remove inconsistencies and is enriched with additional railway information from official PDFs and structured datasets. This enriched dataset is then passed through an embedding model, such as SentenceTransformer, which converts textual information into vector embeddings. These embeddings capture the text's semantic meaning, facilitating efficient retrieval in response to user queries.

The embedding storage and indexing mechanism is implemented using FAISS (Facebook AI Similarity Search), which optimizes the retrieval process by allowing fast vector similarity searches. FAISS indexes the generated embeddings using Approximate Nearest Neighbors (ANN) techniques, ensuring low-latency searches for real-time query resolution.

When a user submits a query, the query handling and response generation module converts it into a vector representation using the matching embedding model. This vector is then matched against the indexed embeddings in FAISS to retrieve the most relevant real-time train information. The retrieved data is then passed into the Mistral AI LLM, a high-performance large language model known for its efficiency and contextual understanding. The system prompt for Mistral AI is structured to condition the model to specialize in Indian Railways, ensuring that the responses produced are precise, domain-specific, and contextually appropriate.

Finally, the output and user interaction module formats the

response in a structured manner for easy interpretation by the end user. The system continuously improves its accuracy by leveraging user feedback, refining the retrieval and response generation process over time. This real-time

VIII. RESULTS AND DISCUSSION

1. Querying Railway Zone Information

When asked about the railway zone of a specific station, such as *Roha*, the system provides a structured response containing:

Query:

```
Enter your query: ROHA is in which Railway zone?
```

Figure 4.1(a) : Query 1

Generated Response:

```
Generated Response:  
ROHA is in the Central Railway (CR) zone.
```

Figure 4.1(b) : Response 1

Similarly, when queried about all railway stations within a particular zone (e.g., *Konkan Railway Zone*), the system retrieves and lists relevant stations along with their station codes and railway zones.

Query:

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

Figure 4.2(a) : Query 2

Generated Response:

```
Generated Response:  
The railway stations in the KR (Konkan Railway) zone include:  
1. KUMTA (KT)  
2. VEER (VEER)  
3. KUDAL (KUDL)  
4. KHED (KHED)  
5. KALAMBANI (KLBN)  
6. INDAPUR (INP)  
7. KAMATHE (KMAH)  
8. KHAREPATAN (KRPN)  
9. KOLAD (KOL)  
10. KANKAVALI (KKW)
```

Figure 4.2(b) : Response 2

2. Real-Time Train Status Retrieval

When asked about the status of a specific train, such as *Maharashtra Splendour*, the system identifies and returns the top 3-5 relevant documents from the train status JSON file. The retrieval is based on a similarity threshold, ensuring that only the most relevant documents are selected for further processing.

Query:

RAG pipeline significantly enhances railway information retrieval, ensuring users receive up-to-date insights while maintaining a scalable and adaptable framework for future extensions into broader transportation domains.

```
Enter your query: Tell me about the time status of  
MAHARASHTRA SPLENDOU
```

Figure 4.3(a) : Query 3

Generated Response:

Generated Response:

Based on your query 'Tell me about the time status of MAHARASHTRA SPLENDOU', here is the relevant information:

- Timestamp: 2024-09-26 14:54:01, Train No: 00189, Train Name: MAHARASHTRA SPLENDOU, Status: ARRIVED, Station: VERNA, Time: 12:45, Delay: -6:00
- Timestamp: 2024-09-26 14:51:01, Train No: 00189, Train Name: MAHARASHTRA SPLENDOU, Status: ARRIVED, Station: VERNA, Time: 12:45, Delay: -6:00
- Timestamp: 2024-09-26 14:54:01, Train No: 00189, Train Name: MAHARASHTRA SPLENDOU, Status: ARRIVED, Station: VERNA, Time: 12:45, Delay: 4:12
- Timestamp: 2024-09-26 14:51:01, Train No: 00189, Train Name: MAHARASHTRA SPLENDOU, Status: ARRIVED, Station: VERNA, Time: 12:45, Delay: 4:12
- Timestamp: 2024-09-26 14:54:01, Train No: 00189, Train Name: MAHARASHTRA SPLENDOU, Status: ARRIVED, Station: VERNA, Time: 12:45, Delay: 6:32
- Timestamp: 2024-09-26 14:51:01, Train No: 00189, Train Name: MAHARASHTRA SPLENDOU, Status: ARRIVED, Station: VERNA, Time: 12:45, Delay: 6:32

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

Figure 4.3(b) : Response 3

3. Complex Query Handling

The integration of Mistral AI significantly enhances the system's ability to handle complex queries. For example, when asked about boarding a train from *Roha*, the system intelligently suggests that the user can take the *LTT-ROHA Passenger* train from *Thane*, providing a structured and logical response rather than listing matching documents.

Query:

```
Enter your query: I want to reach ROHA at the earliest.  
Give me the train I need to catch.
```

Figure 4.4(a) : Query 4

Generated Response:

Generated Response:

To reach ROHA at the earliest, you need to catch the **Diva Sawantwadi Road Passenger (50105)**. This train is scheduled to depart from Dadar Central (DR) at 10:15 PM and will arrive at ROHA (ROHA) at approximately 1:30 AM the next day. This is the fastest and most direct option available. Please confirm your starting station or any other preferences for a more tailored recommendation.

Figure 4.4(b) : Response 4

4. Real-Time Train Status Query

For one of the most critical real-time queries—"What is the current status of <train name>?"—the system retrieves the latest train status, including:

1. Train Number
2. Last Crossed Station with Time
3. Delay Status (if any)

This ensures that users receive up-to-date and actionable information about their train journeys.

Query:

Enter your query: What is the current train status of CSMT-KRMI SPL?

Figure 4.5(a) : Query 5

Generated Response:**Generated Response:**

The current train status of CSMT-KRMI SPL (Train No: 01151) is LEFT, with the most recent update showing it departed from THIVIM station at 13:24 with a delay of 1:1. This information is based on the timestamp from 2025-01-04 13:42:01.

Figure 4.5(b) : Response 5

1. Hugging Face Response:

- a. Primarily performs similarity search, returning all documents that meet the similarity threshold.
- b. The responses lack logical structuring and do not generate human-like replies.
- c. For example, if asked "Hello, how are you?", it fails to recognize the query and throws an error.

2. Hugging Face + Mistral AI Response:

- a. Generates more natural, structured, and contextual responses.
- b. Combines retrieved knowledge with reasoning to form meaningful answers.
- c. Handles conversational queries like "Hello, how are you?" with an appropriate greeting, demonstrating a more interactive and intelligent system.

The integration of Mistral AI enhances the contextual understanding, logical reasoning, and conversational capabilities, bridging the gap between raw similarity search and intelligent response generation. This hybrid approach enables the RAG system to think before answering, making it a more human-like, adaptive, and efficient solution for real-time railway information retrieval.

IX. FUTURE WORK

The development of an interactive, user-driven UI using Streamlit will begin with understanding user requirements to gather insights on design preferences and functionality needs. Once these are defined, we will create an intuitive, engaging UI with interactive elements, followed by extensive testing and refinement based on usability feedback. Feedback handling will be crucial in this process, where feedback will be collected through various channels like in-app forms, surveys, and live chat, categorized into key areas like user experience, performance, and features. Feedback analytics will help prioritize updates, while user support ensures issues are addressed promptly. Additionally, the app will support multilingual operations to cater to a global audience. This includes handling multilingual data input, identifying target languages based on user demographics, and ensuring accurate text translation and localization through collaboration with native speakers. By focusing on these areas, we will make the app accessible to users from different regions, providing a seamless experience and addressing cultural nuances in the design and content.

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/ijse/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 Rodrigues, 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

[18] Gresha Bhatia , Rohini Temkar , Aryan Raje , Arya Raje, Ishita Marathe , Prasad Lahane, “Intelligent Railways: Leveraging Retrieval-Augmented Generation for Smarter Systems”, Communications on Applied Nonlinear Analysis, ISSN: 1074-133X Vol 32 No. 3s (2025), 17 November, 2024

d. Plagiarism report Paper II

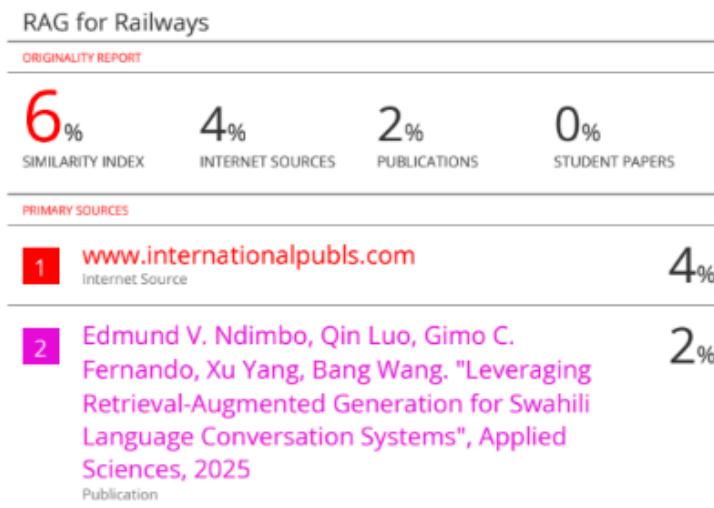


Fig.16.PLAGIARISM REPORT OF PAPER 2

e. Project review sheet 1

Inhouse/ Industry / Innovation/Research: Inhouse												Class: D17 A/B/C			
Sustainable Goal: 9 : Industry, Innovation & Infra												Project Evaluation Sheet 2024 - 25			
Title of Project: RAG for Railways.												Group No.: 7			
Group Members: Aryan Raje (51), Anya Raje (50), Ishita Marathe (41), Prasad Lahane (36)															
Engineering Concepts & Knowledge	Interpretation of Problem & Analysis	Design / Prototype	Interpretation of Data & Dataset	Modern Tool Usage	Societal Benefit, Safety Consideration	Environment Friendly	Ethics	Team work	Presentation Skills	Applied Engg&Mgmt principles	Life - long learning	Professional Skills	Innovative Approach	Research Paper	Total Marks
(5)	(5)	(5)	(3)	(5)	(2)	(2)	(2)	(2)	(2)	(3)	(3)	(3)	(3)	(5)	(50)
5	5	5	3	5	2	2	2	2	2	3	2	2	3	4	47
Comments: Evaluation measures to be undertaken - Response Time, Accuracy, Comparison with existing system.												Dr. Gaurav Bhatia Name & Signature Reviewer 1			
Engineering Concepts & Knowledge	Interpretation of Problem & Analysis	Design / Prototype	Interpretation of Data & Dataset	Modern Tool Usage	Societal Benefit, Safety Consideration	Environment Friendly	Ethics	Team work	Presentation Skills	Applied Engg&Mgmt principles	Life - long learning	Professional Skills	Innovative Approach	Research Paper	Total Marks
(5)	(5)	(5)	(3)	(5)	(2)	(2)	(2)	(2)	(2)	(3)	(3)	(3)	(3)	(5)	(50)
5	5	5	3	5	2	2	2	2	2	3	2	2	3	4	47
Comments: Need some evaluation measure												Pradyumn Satat Jadhav Name & Signature Reviewer 2			
Date: 1st March, 2025															

f. Project review sheet 2

Inhouse/ Industry / Innovation/Research: Inhouse												Class: D17 A/B/C			
Sustainable Goal: 9 : Industry, Innovation and Infrastructure												Project Evaluation Sheet 2024 - 25			
Title of Project: RAG for Railways												Group No.: 7			
Group Members: Prasad Lahane (36), Ishita Marathe (41), Anya Raje (50), Aryan Raje (51)															
Engineering Concepts & Knowledge	Interpretation of Problem & Analysis	Design / Prototype	Interpretation of Data & Dataset	Modern Tool Usage	Societal Benefit, Safety Consideration	Environment Friendly	Ethics	Team work	Presentation Skills	Applied Engg&Mgmt principles	Life - long learning	Professional Skills	Innovative Approach	Research Paper	Total Marks
(5)	(5)	(5)	(3)	(5)	(2)	(2)	(2)	(2)	(2)	(3)	(3)	(3)	(3)	(5)	(50)
5	5	5	3	5	2	2	2	2	2	2	2	2	3	4	47
Comments:												Dr. G. Bhatia Name & Signature Reviewer 1			
Engineering Concepts & Knowledge	Interpretation of Problem & Analysis	Design / Prototype	Interpretation of Data & Dataset	Modern Tool Usage	Societal Benefit, Safety Consideration	Environment Friendly	Ethics	Team work	Presentation Skills	Applied Engg&Mgmt principles	Life - long learning	Professional Skills	Innovative Approach	Research Paper	Total Marks
(5)	(5)	(5)	(3)	(5)	(2)	(2)	(2)	(2)	(2)	(3)	(3)	(3)	(3)	(5)	(50)
05	05	05	03	05	02	02	02	02	02	02	03	02	03	04	47
Comments:												Priyanka Jadhav Name & Signature Reviewer 2			
Date: 1st April, 2025															