

# DPR Analyser System: Automating Detailed Project Report Analysis using RAG and Large Language Models

Nikshap Jain<sup>1</sup>, Dr. Archana Kumar<sup>2</sup>

<sup>1</sup> Scholar of B. Tech 4th Year, Department of Artificial Intelligence & Data Science, Dr. Akhilesh Das Gupta Institute of Professional Studies, Delhi, India..

<sup>2</sup> Professor, Department of Artificial Intelligence & Data Science, Dr. Akhilesh Das Gupta Institute of Professional Studies, Delhi, India.

<sup>1</sup> [nikshapjain@gmail.com](mailto:nikshapjain@gmail.com)

<sup>2</sup> [profdrarchanakumar@gmail.com](mailto:profdrarchanakumar@gmail.com)

## ABSTRACT

Urban planning and infrastructure projects heavily rely on Detailed Project Reports (DPRs), which are notoriously dense and span hundreds of pages. Traditional methods of reviewing these documents are manual, time-consuming, and prone to oversight. The proposed DPR Analyser System is an AI-driven, web-based application designed to streamline the review and extraction of critical insights from these comprehensive reports. Utilizing Retrieval-Augmented Generation (RAG) architecture powered by the OpenAI Assistants API, the system allows users to upload PDF documents and query them directly using natural language. The system provides highly accurate, citation-backed responses and features an intelligent fallback mechanism that triggers internet-based searches when document-specific information is insufficient. Developed using Vue.js on the frontend and Ruby on Rails on the backend, the system ensures seamless communication, robust JWT authentication, and persistent conversation tracking. This paper demonstrates how modern AI frameworks and decoupled web architectures can drastically reduce manual effort and improve decision-making in large-scale project management.

## Abbreviations

AI: Artificial Intelligence

API: Application Programming Interface

DPR: Detailed Project Report

JWT: JSON Web Token

LLM: Large Language Model

NLP: Natural Language Processing

PDF: Portable Document Format

RAG: Retrieval-Augmented Generation

SPA: Single Page Application

## 1. INTRODUCTION

In the domains of urban planning, civil engineering, and government administration, efficient management of Detailed Project Reports (DPRs) is essential. These foundational blueprints detail the planning, financial allocations, and technical specifications of proposed large-scale initiatives. Traditionally, evaluating these extensive PDF documents requires exhaustive manual review by subject matter experts. This process is slow, highly inefficient, and often leads to delays in project approval pipelines.

The DPR Analyser System addresses these challenges by introducing a centralized, intelligent platform for document interrogation. By leveraging Large Language Models (LLMs) and Natural Language Processing (NLP), the system allows stakeholders to upload complex reports and ask targeted questions. Unlike standard search tools, this system uses

Retrieval-Augmented Generation (RAG) to ensure responses are contextually accurate and backed by direct citations from the text.

Furthermore, the system mitigates the limitations of isolated document analysis by incorporating a web-search fallback mechanism. Developed with a decoupled architecture utilizing Vue.js, Ruby on Rails, and PostgreSQL, the application provides a scalable, secure, and user-friendly solution, ultimately aiming to enhance transparency, reduce analytical bottlenecks, and foster smarter infrastructural governance.

## **2. LITERATURE REVIEW**

The transition from physical document storage to digital document management systems (DMS) marked the first major advancement in enterprise efficiency. Early digital systems facilitated basic keyword searching within PDFs but completely lacked contextual and semantic understanding. With the advent of machine learning, NLP techniques were applied to classify documents and extract named entities. Recently, the introduction of Retrieval-Augmented Generation (RAG) has transformed how humans interact with unstructured text data. RAG systems combine the reasoning capabilities of LLMs with specific, retrieved knowledge bases, effectively mitigating the 'hallucination' problem common in standard generative models.

Despite these advancements, many commercial chat-to-PDF applications present significant gaps for specialized enterprise use. Many lack a reliable fallback mechanism; if a query cannot be answered by the uploaded text, the system fails rather than gathering external context. Furthermore, public AI tools often compromise data security, making them unsuitable for confidential infrastructure DPRs. The proposed DPR Analyser System addresses these limitations by providing a custom-built, secure platform that integrates citation-backed RAG querying with intelligent internet search fallbacks.

### **2.1 Objectives**

The primary objective of the DPR Analyser System is to develop a web-based, AI-driven platform that significantly reduces the time required to extract actionable insights from dense project reports. The system aims to simplify document analysis by providing a conversational chatbot interface. A critical objective is to ensure high accuracy and trust by generating citation-backed responses pointing to the exact document sections. Additionally, the system is designed to provide seamless external context via a web-search fallback mechanism, all while maintaining rigorous data security and session persistence through robust backend technologies.

## **3. METHODOLOGY**

The architecture of the DPR Analyser System follows a modern, decoupled client-server model to ensure high scalability, rapid response times, and secure API integration. The system is divided into three primary layers: the presentation layer, the application layer, and the AI/data layer.

The presentation layer is developed using Vue.js, chosen for its reactive data binding and component-based architecture. This layer provides an interactive Single Page Application (SPA) where users can securely upload PDFs, view parsed document metadata, and interact with the chatbot interface in real-time.

The application layer is implemented using Ruby on Rails. This robust backend manages RESTful API endpoints, handles database operations, and orchestrates the complex logic required for the RAG pipeline. It handles JSON Web Token (JWT) issuance and validation, ensuring that all user sessions and document access remain highly secure.

The AI and data layer utilizes PostgreSQL for relational data storage (managing users, documents, and chat histories) and integrates directly with the OpenAI Assistants API. When a query is initiated, the backend formats the prompt, attaches the specific document ID context, and parses the returned stream to verify citations before transmitting the final payload back to the frontend client.

### ***3.1 Data Collection and Preprocessing***

For the development and testing phases, data was sourced from public infrastructure DPRs, environmental impact reports, and government planning PDFs. The system processes these documents using PDF parsing libraries to extract raw textual data. The text is subsequently chunked, embedded, and indexed by the OpenAI API for efficient retrieval. User interaction data, including chronologically sequenced prompts and AI completions, is collected and stored securely in the PostgreSQL database to maintain session state and persistent history.

### ***3.2 Evaluation and Testing***

System evaluation focused on retrieval accuracy, API latency, and architectural reliability. Unit testing was performed on backend Ruby on Rails models and service objects to ensure JWT generation and parsing logic functioned flawlessly. Integration testing validated the asynchronous data flow between the Vue.js frontend and the Rails API, particularly during document uploads and chat polling. Specialized test cases were designed to evaluate the AI engine, including querying specific factual data to verify citation accuracy, and posing unrelated queries to confirm the successful triggering of the web-search fallback module.

### ***3.3 Implementation Workflow***

The operational workflow of the DPR Analyser System proceeds as follows:

1. Authentication: Users securely register and log in, receiving a JWT for session management.
2. Document Upload: The user uploads a PDF. The backend validates the file, stores it, and registers it with the OpenAI API.
3. Query Submission: The user inputs a natural language question via the Vue.js interface.
4. AI Processing & Routing: The Rails backend sends the query to the LLM. If the answer exists in the document, it applies RAG. If not, it triggers the fallback web-search API.
5. Response Generation: The system returns a formatted response complete with exact citations and source links.
6. State Persistence: The conversation is saved to the database for future reference.

## **4. RESULTS AND DISCUSSION**

The DPR Analyser System was successfully implemented as a highly functional, enterprise-grade prototype. Testing demonstrated a radical acceleration in document review times. Users successfully uploaded multi-hundred-page project reports and received accurate, context-aware answers within seconds. The citation feature proved highly effective, successfully pinpointing the exact sections of the document to foster trust in the AI's outputs.

Furthermore, the intelligent fallback mechanism functioned seamlessly. When posed with queries regarding external variables (such as current macroeconomic rates not explicitly stated in an older DPR), the system successfully retrieved, synthesized, and integrated real-time information from the web into the chat thread.

The Vue.js frontend maintained a lightweight footprint, ensuring smooth rendering of chat elements, while the Ruby on Rails backend managed concurrent API requests efficiently. The results clearly indicate that integrating modern web frameworks with advanced LLMs significantly mitigates the manual bottlenecks of traditional document analysis.

## **5. CONCLUSION AND FUTURE WORK**

The DPR Analyser System demonstrates the transformative potential of artificial intelligence in the realm of urban planning and infrastructure management. By leveraging Ruby on Rails, Vue.js, and the OpenAI Assistants API, the project achieved its goal of building a smart, secure platform that automates the tedious process of reviewing Detailed Project Reports. Features such as citation-backed responses and intelligent web-search fallbacks effectively address the limitations of standard generative models, providing a reliable tool that improves efficiency, accuracy, and decision-making speed.

Future enhancements will focus on integrating multi-modal AI capabilities, enabling the system to interpret visual data, architectural blueprints, and engineering diagrams embedded within the reports. Expanding the architecture to support

cross-document querying—allowing users to compare multiple DPRs simultaneously—will add significant value. Additionally, direct integration with internal government databases could automate the cross-referencing of proposed project data against existing municipal infrastructure records.

## 6. REFERENCES

- [1] Sommerville, I. (2016). Software Engineering, 10th Edition, Pearson Education.
- [2] Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach, McGraw-Hill.
- [3] Lewis, P. et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems.
- [4] Vue.js Team. (2024). Vue.js Documentation. Available: <https://vuejs.org/guide/introduction.html>
- [5] Ruby on Rails Core Team. (2024). Ruby on Rails Guides. Available: <https://guides.rubyonrails.org/>
- [6] OpenAI. (2024). OpenAI Assistants API Documentation. Available: <https://platform.openai.com/docs/assistants/overview>
- [7] Internet Engineering Task Force. (2015). JSON Web Token (JWT) Specification. Available: <https://jwt.io/>