SYSTEM ANALYSIS

This chapter provides an in-depth analysis of the existing and proposed systems, highlighting the limitations of current language model solutions and the advancements introduced by the project. It also outlines the tools, technologies, hardware, and software specifications essential for the system's development. By examining the frameworks, libraries, and services utilized, this chapter lays the groundwork for understanding the design and implementation phases of the project.

2.1 EXISTING SYSTEM

Existing systems, such as standard Large Language Models (LLMs), primarily rely on parametric knowledge encoded during their pre-training phase. These models generate responses based on patterns and knowledge learned from vast datasets but often fail to incorporate real-time, context-specific information. This limitation leads to inaccurate or irrelevant outputs, especially when the model's pre-trained knowledge does not align with the user’s query.

To address this, Retrieval-based systems integrated with language models have been developed. These systems fetch external knowledge from a database or context store, providing relevant context to complement the language model’s responses. However, such systems are vulnerable to the quality and integrity of the retrieved contexts. When the database contains corrupted, malicious, or low-quality data, the language model may incorporate these flawed inputs into its responses, resulting in misinformation or low-quality outputs.

The limitations of these existing systems highlight the need for more robust solutions to improve context relevance, mitigate the influence of corrupted data, and ensure responses align with the user’s query.

2.2 PROPOSED SYSTEM

The proposed system addresses the limitations of existing language model-based solutions by introducing a robust, a module designed to enhance the accuracy and reliability of generated responses.

The ROBUST RAG module focuses on extracting keywords from the retrieved contexts and assigning scores to prioritize high-quality, information-rich contexts. By scoring potential contexts, the system effectively filters out corrupted or low-quality data, significantly reducing the likelihood of generating inaccurate responses. This ensures that only the most relevant and trustworthy contexts contribute to the final output.

The system also possesses additional features with models such as the first one, UPLOAD DOCUMENTS, enables users to store their documents in the specified storage, while the other model, RETRIEVE, assists in fetching the top k most relevant contexts based on the user-provided prompt.

To support scalability and flexibility, the system integrates with cloud services like GOOGLE CLOUD and AMAZON WEB SERVICES (AWS) and utilizes vector databases such as CHROMA for document storage, enabling users to store and retrieve data according to their specific requirements which is done using upload documents model and retrieve model.

Furthermore, the system provides seamless integration with advanced language models such as AWS CLAUDE SONNET, Azure's OPENAI, and Google's GEMINI, allowing users to select their preferred model based on their needs in generating responses which is utilised in retrieve and generate model and in reliable retrieve and generate model.

2.3 PROJECT INITIATION

The initiation phase of this project focused on addressing the critical limitations of language models when handling corrupted or low-quality data and their reliance on pre-trained knowledge over provided contexts. The goal was to develop a robust system capable of extracting and prioritizing high-quality, relevant contexts and also ensuring the language model’s responses remain accurate and contextually grounded. The tools decided to use are LangChain, Streamlit, and Chroma Vector Database. Resources such as AWS Elastic Compute Cloud Servers, and GPU-powered computation platforms like Google Collaboratory were identified to support the project's technical requirements. Design to implement the keyword extraction and scoring methods for robust module and the secondary module to align responses to retrieved contexts are established, ensuring the system's goals were well-defined.

2.4 SYSTEM SPECIFICATIONS

This section outlines the essential hardware and software specifications required for the successful development and deployment of the project. It includes detailed configurations of computing resources, storage solutions, programming tools, and cloud-based services, ensuring optimal performance and scalability for the system.

2.4.1 Hardware Specifications

The table 2.1 outlines the hardware resources utilized in the project, detailing the system configuration, including processor specifications, memory capacity, storage type, and disk size. It also highlights cloud-based resources such as storage services, and servers, which provide the computational power and scalability necessary for efficient development and execution.

Table 2.1 Hardware Specifications

Component	Configuration
System Type	64-bit Operating System
Processor	AMD Ryzen 7 7435HS
Central Processing Unit (CPU) Cores	8
Random Access Memory (RAM)	24 GB
Solid State Drive (SSD) Storage	512 GB
Cloud-based Resources	Google Cloud Storage (GCS), Amazon Simple Storage Service (Amazon S3) Bucket
Cloud-based Servers	Amazon Elastic Compute Cloud (Amazon EC2) Servers

2.4.2 Software Specifications

The table 2.2 lists the key software components utilized in the project, including the operating systems, development environments, programming language, APIs, storage services, version control tools, and vector databases.

Table 2.2 Software Specifications

Component	Configuration
Operating System (OS)	Windows 11 Home, AWS Linux Instances
Integrated Development Environment (IDE)	Google Collaboratory, Visual Studio Code (VS Code)
Programming Language	Python
Application Programming Interface (API) Services	Google Cloud Discovery Engine
Storage Services	Amazon Simple Storage Service (Amazon S3) Bucket, Google Cloud Storage
Version Control	Git and Github
Vector Databases	Chroma

2.5 TOOLS AND TECHNOLOGIES USED

The tools and technologies employed in this project are listed below in detail that encompass a range of software frameworks, libraries, and cloud services essential for developing a robust and scalable system.

2.5.1 Integrated Development Environment (IDE)

An IDE provides a unified interface with tools like code editors, debuggers, and execution environments to streamline the development process. For this project, Google Colab is used for Python execution with TPU support, and Visual Studio Code is employed for efficient local development and debugging.

2.5.1.1 Visual Studios Code (VS Code)

A lightweight, versatile code editor offering extensive features like debugging, syntax highlighting, and other support for efficient software development. The Upload Document, Retrieve, Retrieve and generate, Reliable retrieve and generate models are developed and tested using this IDE which is integrated with git and Python.

2.5.1.2 Google Collaboratory (Google Colab) Notebook

A cloud-based notebook platform for Python programming that provides free access to Graphics Processing Unit (GPU) and Tensor Processing Unit (TPU), ideal for machine learning and data analysis. The Context-aware Decoding model is developed and tested using this IDE to utilise TPUs to processes the tensor operations.

2.5.2 Vector Database - Chroma

Vector databases are optimized for storing and retrieving high-dimensional vector data, enabling efficient similarity searches and AI-driven applications.

In this project, Chroma is used for context storing and retrieval to enhance data relevance and accuracy. Chroma is a vector database optimized for storing and retrieving embeddings, enabling efficient similarity searches in applications like semantic retrieval.

2.5.3 Storage Services

Storage services provide scalable and secure solutions for storing and managing data in the cloud. This project utilizes Amazon S3 Bucket and Google Cloud Storage to handle document storage and retrieval efficiently.

2.5.3.1 Amazon Simple Storage Service (Amazon S3) Bucket

A secure, scalable object storage service offered by Amazon Web Services, suitable for storing large volumes of unstructured data.

2.5.3.2 Google Cloud Storage (GCS) Bucket

A fully managed, reliable storage solution provided by Google Cloud, offering multi-region storage with robust data security features.

2.5.4 Programming Language - Python

A programming language is a formal set of instructions used to communicate with computers and develop software applications.

This project is developed purely using Python Programming Language. Python is a high-level, versatile programming language widely used in data science, web development, and AI for its simplicity and extensive library ecosystem [3].

2.5.5 Application Programming Interface (API) Services - Google Cloud Discovery Engine

API services enable applications to communicate and exchange data seamlessly through predefined protocols and interfaces.

Google Cloud Discovery Engine is a cloud-based API service that enhances search and discovery within applications by delivering relevant and personalized results using machine learning.

2.5.6 Frameworks

A framework is a structured platform that provides pre-built components and guidelines to streamline the development of applications.

2.5.6.1 LangChain

LangChain is a framework designed to develop applications powered by large language models (LLMs) through seamless integration of context retrieval and

response generation. It simplifies building advanced AI systems like chatbots and agents by connecting LLMs to external data and APIs.

2.5.6.2 Streamlit

Streamlit lets developers transform Python scripts into interactive web apps in minutes. Build dashboards, generate reports, or create chat apps.

2.5.7 Libraries

A library is a collection of pre-written code modules that provide reusable functionalities to simplify specific tasks in development. In this project, libraries like spaCy, and Boto3 are used for tasks like language processing, deep learning, and cloud service integration.

2.5.7.1 Boto3

Boto3, the AWS SDK for Python, is used to create, configure, and manage AWS services, such as Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3).

2.5.7.2 spaCy

spaCy is a library for advanced Natural Language Processing in Python and Cython. It features state-of-the-art speed and neural network models for tagging, parsing, named entity recognition, text classification and more.