PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013
Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

PGRKAM SMART ASSISTANT

A PROJECT REPORT

Submitted by

ASHWIN R - 20221CAI0042

SACHIN S - 20221CAI0026

HARSHIT B- 20221CAI0017

Under the guidance of,

Mr. JAI KUMAR B

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

PRESIDENCY UNIVERSITY

BENGALURU

DECEMBER 2025

PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013
Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this report, "PGRKAM SMART ASSISTANT," is a bonafide work of "ASHWIN R (20221CAI0042), SACHIN S (20221CAI0026), and HARSHIT B (20221CAI0017)," who have successfully carried out the project work and submitted the report for partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE ENGINEERING, ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING during 2025-26.

Dr. Jai Kumar B
Project Guide
PSCS
Presidency University

Ms. Suma N G
Program Project Coordinator
PSCS
Presidency University

Dr. Sampath A K
Dr. Geetha A
School Project Coordinators
PSCS
Presidency University

Dr. Zafar Ali Khan N
Head of the Department
PSCS
Presidency University

Dr. Shakkeera L
Associate Dean
PSCS
Presidency University

Dr. Duraipandian N
Dean
PSCS & PSIS
Presidency University

Name and Signature of the Examiners

S.No	Name	Date	Signature
1.	Jijina M T	4/12/25	
2.	Shana Aneevan	4/12/25	

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, the students of the final year of B.Tech in COMPUTER SCIENCE ENGINEERING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING) at Presidency University, Bengaluru, named ASHWIN R, SACHIN S, and HARSHIT B, hereby declare that the project work titled **"PGRKAM Smart Assistant"** has been independently carried out by us and submitted in partial fulfillment for the award of the degree of B.Tech in COMPUTER SCIENCE ENGINEERING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING) during the academic year of 2025-26. Further, the matter embodied in the project has not been submitted previously by anybody for the award of any degree or diploma to any other institution.

ASHWIN R	20221CAI0042	
SACHIN S	20221CAI0026	
HARSHIT B	20221CAI0017	

PLACE: BENGALURU

DATE: 4/12/2025

ACKNOWLEDGEMENT

For completing this project work, we/I have received the support and the guidance from many people whom I would like to mention with a deep sense of gratitude and indebtedness. We extend our gratitude to our beloved **Chancellor, Pro-Vice Chancellor, and Registrar** for their support and encouragement in the completion of the project.

I would like to sincerely thank my internal guide, Mr. Jai Kumar B, Assistant Professor, Presidency School of Computer Science and Engineering, Presidency University, for his moral support, motivation, timely guidance, and encouragement provided to us during the period of our project work.

I am also thankful to **Dr. Zafar Ali Khan N, Professor and Head of the Department, Presidency School of Computer Science and Engineering,** Presidency University, for his mentorship and encouragement.

We express our cordial thanks to **Dr. Duraipandian N**, Dean of PSCS & PSIS; **Dr. Shakkeera L**, Associate Dean of the Presidency School of Computer Science and Engineering; and the Management of Presidency University for providing the required facilities and intellectually stimulating environment that aided in the completion of my project work.

We are grateful to **Dr. Sampath A. K. and Dr. Geetha A., PSCS** Project Coordinator; and **Ms. Suma N G, Program Project Coordinator**, Presidency School of Computer Science and Engineering, for facilitating problem statements, coordinating reviews, monitoring progress, and providing their valuable support and guidance.

We are also grateful to the teaching and non-teaching staff of Presidency School of Computer Science and Engineering and also staff from other departments who have extended their valuable help and cooperation.

ASHWIN R

SACHIN S

HARSHIT B

Abstract

The Punjab Government launched the Punjab Ghar Ghar Rozgar and Karobar Mission (PGRKAM) to provide people with access to a wide variety of job opportunities (both government jobs and private sector) through a single digital platform. The application allows users from rural and semi-urban communities to register for job opportunities via the Internet, but there are many barriers for these users in accessing the site, including limitations in their knowledge of English and/or Punjabi; a lack of digital literacy; challenges in navigating the website; and waiting for updates regarding the status of their applications.

Conversational AI has the potential to help reduce these barriers and improve access to job opportunities and government or private schemes for users from rural and semi-urban communities. The PGRKAM project includes a Smart Assistant that can interact with users in multiple languages (Punjabi and English) to provide real-time information about jobs, schemes, and training programs through conversations. The Smart Assistant has been built using the FastAPI web framework with an RAG (Retrieval Augmented Generation) architecture, which provides for the processing of user requests.

The Smart Assistant uses Natural Language Understanding (NLU) and a hybrid retrieval approach to process requests. In addition to being able to accept voice-based input (using Speech-to-Text) and providing real-time translation capabilities, the Smart Assistant provides a high level of usability for users in rural and semi-urban communities.

Table of Content

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
ASR	Automated Speech Recognition
BM25	Best Matching 25
CSV	Comma Separated Values
DB	Database
FAISS	Facebook AI Similarity Search
FAQ	Frequently Asked Questions
GIS	Geographic Information System
GPU	Graphics Processing Units
GSV	Gross State Value
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol Secure
ICT	Information and Communication Technology
IR	Information Retrieval
IT	Information Technology
JSON	JavaScript Object Notation
LLM	Large Language Model
ML	Machine Learning
NER	Named Entity Recognition
NLP	Natural Language Processing
NLU	Natural Language Understanding
NMT	Neural Machine Translation
PDF	Portable Document Format

PGRKAM	Punjab Ghar Ghar Rozgar and Karobar Mission
PII	Personally Identifiable Information
PSA	PGRKAM Smart Assistant
QA	Question Answering
RAG	Retrieval Augmented Generation
RAGAS	Retrieval-Augmented Generation Assessment
REST	Representational State Transfer
RRF	Reciprocal Rank Fusion
SAAS	Software as a Service
SDG	Sustainable Development Goals
UI	User Interface
WER	Word Error Rate

Chapter 1
Introduction

Digital employment portals are critical platforms that connect job seekers to employment opportunities, both in the government and private sectors. However, many citizens, particularly those in rural and semi-urban settings, are unable to use these portals due to language barriers, poor digital skills, and rigid filtering criteria. The PGRKAM portal is a major provincial initiative to create jobs in Punjab but, despite its benefits, many users find it complex and still struggle to find information. The PGRKAM Smart Assistant Chatbot intends to fill in the gaps that traditional user support could not provide and provides an intelligent, multilingual, speech-enabled user support system that can understand queries and provide job information quickly.

1.1 Background

The PGRKAM (Punjab Ghar Ghar Rozgar and Karobar Mission) portal and mobile application were created by the Government of Punjab to serve a centralized platform for job seekers and employers. The portal collects private sector jobs, public sector job postings, self-employment, jobs abroad, education opportunities, counseling, armed forces induction, and job expos, making this a necessary digital infrastructure for employability and career services in Punjab.

Despite this extensive functionality, the PGRKAM platform still requires users engage in traditional web navigation: users must ultimately manually navigate through several menus and sub-menus, links, and pages to find the information they are looking for. For first-time users, the rural youth, and those with low digital self-efficacy, manually navigating the information can be tedious, slow, and subject to errors. At the same time, there are rapid developments utilizing NLP, LLM’s and RAG to facilitate conversational systems that understand natural language inquiries, access structured and unstructured knowledge sources, and determine contextual responses relevant to the user's public service needs.

1.2 Statistics of the Project

As of 2024, the Punjab Ghar Ghar Rozgar and Karobar Mission (PGRKAM) portal has successfully registered over 1.1 million job seekers and 8,500 employers, not to mention thousands of private job postings . However, analytics on the platform and extensive survey data indicates that while 800,000+ users have logged on to the portal, many have challenges using the platform in an effective way to get continuous work. The issue originates mostly from a lack of digital literacy, language barriers, and navigation.

A recent survey in rural Punjab found that over 70% of people do not have Information and Communication Technology (ICT) literacy at even the most basic level, limiting access to an online employment program. Punjab currently has a youth unemployment rate of 14.9% on a yearly basis, which is higher than the national average, and over 80% of youth employed do not have job satisfaction. The dissatisfaction of youth employed is usually around major skill mismatches with jobs, along with limited or no access to employment programs. Further, while Punjab has projected Gross State Value (GSV) growth of 6.8%, there remains a digital divide with lower rural internet capacity and gender variance, excluding many from any employment programs

1.3 Prior Existing Technologies

Retrieval-Augmented Generation (RAG) offers a reliable method to ground the output from large language models with external sources of knowledge. Lewis et al. [1] initially proposed a base framework for RAG, which was later expanded upon by Gao et al. [2], who demonstrated that dense retrieval can be completed even when using a zero-shot approach. Although dense retrieval performs poorly on domain-specific data associated with structured regulation, there was a need for a Hybrid Retrieval Architecture for this purpose.

Further research supports the use of Hybrid Search, combining Sparse BM25 with Dense Semantic Vectors, to outperform either Sparse or Dense methods in all areas of Domain Specific RAGs. Sawarkar et al. [3] and Abdulkhader N[4] provide experimental support for the superiority of Hybrid Retrieval over both Sparse and Dense Methods when applied to

policy and governing documents. Rackauckas et al. [5] introduced Reciprocal Rank Fusion (RRF) to accurately combine Sparse and Dense Retrieval Results.

In the domain of NLU (Natural Language Understanding) for low-resource languages, NER (Named Entity Recognition) models that are typically used in a supervised manner are not applicable due to the unavailability of labelled datasets. The Zero-shot NER model [6][7] developed by GLiNER enables users to extract entities without requiring task-specific training.People, such as Zaratiana et al.[6] and Pattanayak et al.[8], have drawn attention to the issue of tokenising Indic Languages, whereas the HiNER[9] dataset has further demonstrated the lack of availability of large-scale NER resources for the Punjabi language.

Multilingual interaction is an important aspect of citizen-centred platforms in the Indian context, and IndicTrans2 [10] enables high-quality translation to and from Punjabi and English. IndicSUPERB[11] benchmarks highlight the challenges that exist in developing ASR systems for Indic Languages, while research on Punjabi language generation[12] and recent improvements to the Whisper ASR solutions[13] for Indian languages justify the need for utilizing both an ASR and a translation pipeline rather than developing fully trained speech models.

Modern public service chatbots require a microservices architecture based on APIs that can be scaled effectively. Based on numerous examples [14][15][16][17], FastAPI provides an excellent foundation for this type of deployment. E-Government studies confirm that conversational systems are an excellent interface for access to public services [15]. The use of AI-driven software architecture [16] and a standardized RAG evaluation framework [18][19][20][[21][22] provides a solid basis for building reliable, scalable, and trustworthy conversational systems.

1.4 Proposed Approach

For this project, we will design and build a multi-lingual interactive job assistant system on the PGRKAM platform, leveraging a Retrieval-Augmented Generation (RAG) framework instead of a keyword-based or form-driven approach. We chose our design to help ensure

that user requests are answered with reliable governmental data and achieve fewer "hallucinations," as indicated in original RAG research [1].

Our goal is for users to interact with the employment application using either natural language via text or voice. The focus of this design will be to enhance the accessibility, usability, and trustworthiness of the application.

To create a system that delivers high accuracy in retrieval within governmental document collections, this project is utilizing a Hybrid Information Retrieval model that combines the use of sparse retrieval (BM25) with dense semantic vector-based retrieval. Research has indicated that a Hybrid model produces better total retrieval accuracy consistently than either [3][4] individually, particularly with domain-specific RAG systems that include structured policy/regulatory documents. In addition, to strengthen the robustness of the ordering of Hybrid RAG systems, we are using Reciprocal Rank Fusion (RRF) to fuse the output of both methods for the RAG system; recent studies indicate that RRF offers better performance than other available fusion techniques [5]. The proposed retrieval method developed will ensure that the user query's semantic meaning and eligibility criteria are represented proportionally.

The proposed method for Natural Language Understanding (NLU) does not require training large supervised models due to the availability of little or no annotated data in Punjabi. Instead, a zero-shot NER approach is used to train generalist models such as GLiNER, which have demonstrated good results in low-resource language environments. This enables the extraction of structured entity types; for example, job role, qualification, location, and age, etc., without retraining for each specific use case. The tokenization and pre-processing challenges for the Indic script are being addressed based on previously established conclusions from Indic NLP research.

The desired user base demographic is mainly Punjabi-speaking, and therefore the system will offer multilingual speech and translation services. The Punjabi/English translation will be performed via modern Indic language translation models. Automatic Speech Recognition (ASR) for voice interaction will be via Whisper technology, and this has already been

successfully validated in Indian language ASR systems. Hence, the new system will allow for both literate and semi-literate users to interact effectively with the system.

For deployment and governance purposes, this proposed architecture uses a microservices-based API-driven architecture using FastAPI to allow for high scalability and reliability while enabling rapid integrations with governmental platforms [14][17]. Additionally, research conducted regarding the implementation of public service chatbots supports the conclusion that such architectures significantly enhance e-governance systems' responsiveness, maintainability, and engagement of citizens with the e-governance systems [15]. The quality and safety of the responses generated will be assessed using conventional RAG evaluation frameworks to ensure their faithfulness, detection of hallucinations, and retrieval accuracy [18][19][20], and extended failure analysis as defined by the latest studies on the reliability of RAG results [22].

1.5 Objectives

The development of the PGRKAM Smart Assistant is guided by the following SMART objectives, aligned with the proposed Hybrid RAG-based multilingual employment assistance framework.

Objective 1: Multi-Language Query Understanding

The design and implementation of a multi-language query understanding module will enable the processing of Punjabi, English, and Code Mixed queries through both text and speech. The system will have high performance for language detection and the interpretation of user intent and entities, including the job role, qualifications, locations, and age, using a zero-shot named entity recognition (NER) approach. A preprocessing pipeline will apply normalization and transliteration handling, and will have an acceptable latency for end-to-end input to support real-time interaction.

Objective 2: Implement a Hybrid Information Retrieval (IR) Structure Combined with RAG-Based Generation of Responses

The development of a Hybrid Information Retrieval pipeline will integrate a BM25 lexical-based search structure with dense vector-based semantic retrieval methods to provide accurate document retrieval from the PGRKAM datasets. The Hybrid RAG framework will allow the system to provide factually correct and citation-supported responses to users, while the main aim is to provide the highest level of accuracy in the retrieval process and to reduce hallucination in all system-generated outputs.

Objective 3: Creation of a Highly Secure and Scalable Backbone

Utilizing FastAPI to create a secure backend capable of processing requests for multiple users at once and providing high-speed API responses with minimal latency. This backend will allow the creation of real-time documents; receive, store, and retrieve documents in both the user and backend format, and deliver an intuitive interface for retrieving information and offering guidance on eligibility.

Objective 4: Develop a User-Intuitive Interface with a Bilingual Chat System

A bilingual (Punjabi/English) chat system will be developed to provide easy access to the information required by users through text or voice. Users with varying levels of digital literacy will benefit from a clearly displayed and concise summary of the documents created and the eligibility for programs, along with follow-up information about the next steps to take regarding the program.

Objective 5: Test the System's Performance and Usability

The system will be tested using a controlled testing approach to determine the performance and usability of the system; how many tasks were successfully completed, the precision of the information retrieved, the quality of the response given, the latency of the retrieval process, and the satisfaction of the user. Additionally, the system will be tested to determine

its scalability as the number of users increases; therefore, preparing the necessary documentation to assist in facilitating future administrative usage and deployment of the system will also be included in this objective.

1.6 SDGs

The 17 SDGs established by the United Nations constitute a global agenda that calls for inclusive social, economic, and technological growth by 2030. Technology-based public service platforms will assist in reaching the SDGs through enhanced access, reduced inequality, and improved governance over digital services. The PGRKAM smart assistant supports the following SDGs:

Figure 1.1 Sustainable development goals

Figure 1.1 illustrates the 17 SDGs defined by the United Nations.

SDG 1 – No Poverty

The PGRKAM Smart Assistant helps achieve SDG 1 by creating opportunities for individuals to have access to wage employment, self-employment schemes, and entrepreneurial opportunities. The language used in the application reduces barriers for socio-economically disadvantaged individuals to access employment opportunities by

enabling them to access the information they need to find out what they qualify for and how to apply without having to struggle through multiple channels. Reduced time in searching for verified employment opportunities increases the chances of developing a sustainable income, thereby decreasing poverty levels.

SDG 4 – Quality Education

The PGRKAM Smart Assistant supports SDG 4 by enabling users of the application to easily locate skill development, vocational training, and educational programs that are available via the PGRKAM resource. The ability to read and understand all information regarding eligibility, deadlines, and documentation in multiple languages provides non-native English speakers with an opportunity to research, evaluate, and select educational programs that meet their needs. The ability to read and understand all information about skill development, vocational training, and education enables non-native English speakers to make better decisions for continuing education.

SDG 8 - Decent Work and Economic Growth

The Smart Assistant helps people connect with job opportunities within the public and private sectors while making it easier for them to find jobs that meet their needs, find jobs quickly, and determine if they are eligible for jobs through government agencies. The Smart Assistant also helps people participate more in labour markets, removes barriers to finding jobs, and helps support inclusive economic growth in Punjab.

SDG 9 - Industry, Innovation and Infrastructure

The project supports SDG 9 by providing AI-based digital infrastructure, multilingual NLP, hybrid RAG, and completely scalable API architecture. The project demonstrates the ability of new technologies to transform traditional government job portals into smart, interactive government services.

SDG 10 - Reduced Inequalities

By providing a voice-enabled, bilingual user interface, the Smart Assistant reduces linguistic, digital, and spatial barriers to participating in government jobs for people in rural, semi-urban, or low-literacy communities. This allows for fairer access for individuals in these areas to government employment and reduces structural inequalities.

1.7 Overview of project report

The PGRKAM Smart Assistant Report describes how the PGRKAM Smart Assistant was designed, developed, implemented, and evaluated. The PGRKAM Smart Assistant is a multilingual, retrieval-augmented conversational system that provides information regarding employment and public services in Punjab.

Chapter 1 introduces the project with a general background that includes a motivational and problem statement, along with objectives, relevance to the SDGs, an overview of existing technologies, and a proposed conceptual approach to the development of the PGRKAM Smart Assistant.

In Chapter 2, an extensive literature review for hybrid retrieval and RAG systems, zero-shot Named Entity Recognition (NER), and multilingual Natural Language Understanding (NLU), Indic language speech and translation technologies, scalable chatbot architectures, and standard evaluation frameworks for RAG systems is presented.

In Chapter 3, the methodology used to develop the system is explained in detail. This includes the system development lifecycle, how to acquire data, how to preprocess the data, and how to develop an overall experimental and testing strategy.

In Chapter 4, the project management issues such as the development plan, schedule, risk analysis, resource allocation, and feasibility assessment associated with implementing the PGRKAM Smart Assistant in a public sector context are discussed.

In Chapter 5, the System Analysis and Architecture covers the functional and non-functional requirements, data flow diagrams, module-level design, and high-level structure of the Hybrid RAG system.

In Chapter 6, the System Implementation focuses on FastAPI back-end architecture, hybrid retrieval methods and architecture of the vector database, multilingual capabilities, and typical usage through the web interface.

In Chapter 7, System Evaluation and Results, outlines the performance evaluation of the system, in terms of retrieval accuracy, response accuracy, the results of the Multilingual Testing performed, latency analyses, and results of controlled user testing.

In Chapter 8, Ethical, Legal, Social, and Sustainability Issues and Security Issues, the ethical, legal, social, environmental, and security aspects of creating an AI employment support system in a government context are discussed.

Finally, Chapter 9, the conclusion, provides a summary of the project, assesses how well the original objectives were met, and identifies possible avenues for future development and research.

Chapter 2

Literature review

2.1 Literatures Reviewed

Lewis et al. (2020) – Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

RAG or retrieval-augmented generation, as proposed by Lewis et al. (2020), is an innovative method that employs a combination of both dense document retrieval methods and transformer generator models to retrieve relevant evidence from large document repositories via a retrieval process. When using dense indexing, the performance of RAG's end-to-end training shows strong performance on knowledge-intensive tasks, such as open-domain question answering. The authors conducted experiments related to these claims, although they do indicate that the performance of RAG is dependent on the quality of retrieval and that significant infrastructure is required to support large-scale dense indexing.

Gao et al., 2023 – Precise Zero-Shot Dense Retrieval without Relevance Labels

Gao et al. (2023) conducted a zero-shot study on the dense retrieval problem without the aid of any labeled relevance data. Their approach involved two techniques for improving the accuracy of dense retrieval methods: using contrastive prompts to contrast retrieval prompts with query representations, and aligning those queries with their respective representations. The findings demonstrate that dense retrieval methods are capable of transferring across datasets without requiring additional training (or fine-tuning) after the datasets have been constructed (i.e., creating dense representations). Furthermore, their approach showed improved performance compared to existing methods of unsupervised baseline dense retrieval methods. Despite this success, dense retrieval continues to have challenges associated with lexical matching, and semantically similar documents were also assessed.

Sawarkar et al., 2024 – Blended RAG with Hybrid Retrievers

Sawarkar et al. (2024) proposed "Hybrid Retrieval Strategies Using BM25 and Semantic Similarity," which examines a combined approach that employs sparse (keyword) and dense (semantic) retrieval methods in order to evaluate the accuracy of both retrieval methods. In order to evaluate this combined approach, the authors conduct a comparison of the performance of each method using a separate analysis conducted on different information types (keyword search versus semantic search), as well as a comparison of various datasets that were developed specifically for use with healthcare domain-specific queries. The authors conclude that when both methods are combined for hybrid retrieval analysis, the results are more accurate and have a higher recall rate than either of the methods individually when examining answers. The authors acknowledge that while both methods produce different types of results, together they create a more powerful document selection ability than either method alone. Furthermore, the authors assert that the combined (hybrid) approach is advantageous for structured and regulatory document retrieval purposes.

Kayamkulam Abdulkhader N, 2025 – Evaluating Retrieval Strategies for Domain-Specific RAG Systems

The researcher, Kayamkulam Abdulkhader N (2025), systematically assessed multiple strategies of reference and retrieval for RAG-style systems by evaluating the relative performance of those strategies (sparse, dense, and hybrid) using datasets developed from government and enterprise documents. By comparing results from Recall@K and Precision@K metrics, the researchers found that hybrid retrieval performed better than all other methodologies. The researchers also found that the types of limitations associated with dense retrieval were less effective due to its inability to accurately identify and handle numeric/categorical constraints, and that sparse retrieval had more limitations due to its inability to identify and handle similar terms and paraphrased terms. This research indicates that hybrid architectures will provide the best combination of precision and recall for systems in a specific domain.

Rackauckas, 2024 – RAG-Fusion and Reciprocal Rank Fusion (RRF)

In proposing the concept of Reciprocal Rank Fusion (RRF), Rackauckas (2024) suggests a method for combining various rankings produced by multiple information retrieval systems. It employs an averaging approach to generate scores based on inverse rank positions and does not require any additional pre-training on the dataset. Results demonstrate that using this method provides significant improvements in both the robustness and stability of ranking when using various datasets. In addition, experimental comparisons showed that RRF provides improved recall in comparison to individual retrieval system rankings. The method of fusing retrieval system rankings is also very efficient in terms of both computation and independence of the models used by each retrieval system. However, RRF itself cannot resolve errors that were made by all three retrieval systems at the same time.

Zaratiana et al., 2024 – GLiNER: Generalist NER Model

Zaratiana et al. (2024) introduced a general-purpose zero-shot named entity recognition (NER) system called GLiNER. With GLiNER, users can utilize natural language prompts to identify named entities within any body of text, rather than creating an explicit training dataset (like most traditional NER systems) for performing this task. In addition to having been evaluated against dozens of NER benchmarks, GLiNER has demonstrated strong performance across multiple domains as well as good model generalizability, without requiring the availability of very large annotated corpora. However, when dealing with highly technical disciplines, GLiNER's overall ability to accurately identify named entities decreases. In this regard, this research highlights the importance of finding a proper balance between the two important considerations related to designing and developing an effective GIS system, which are the flexibility offered by GLiNER and the optimal level of precision achieved.

Sohn et al., 2024 – Zero-Shot NER for Low-Resource Languages

According to the research conducted by Sohn et al. (2024), the authors aim to evaluate the ability of zero-shot named entity recognition (NER) across a range of under-resourced languages. They analyze how transformer-based zero-shot models compare with supervised

models using a limited amount of annotated training data. The findings demonstrate that for annotation resources that are significantly limited in number, past supervised models have performed less effectively than zero-shot models. The research provides insight into how key aspects of cross-lingual transfer learning may contribute to this difference. However, due to greater morphological complexity, morphologically rich languages continue to face challenges in detecting the boundaries of named entities. The authors argue that zero-shot NER is an important method to consider for eventual use in under-resourced language environments.

Pattnayak et al., 2025 – Tokenization Challenges in Indic Languages

Pattnayak et al. (2025) studied how different tokenization strategies for Indic languages affected Zero-Shot Named Entity Recognition (NER). Their study demonstrated that challenges like agglutination, different spellings (orthographic variation), and code-mixing created issues with NER systems for Indic languages. Their research findings indicate that poor tokenization negatively affected the performance of NER systems. They presented subword-based tokenizers and rule-aware tokenizers to address these issues, and through experimentation, they found that both tokenization techniques increased the performance of NER systems across multiple Indic languages; however, tokenization continues to be a challenge for Indic natural language processing pipelines.

Murthy et al., 2022 – HiNER: Large Hindi NER Dataset

Murthy et al. (2022) provide a significant resource in the form of HiNER: the largest collection of Hindi Named Entities to date, which contains a much greater variety of named entities and realistic text sources than any prior dataset. Models trained using this data performed substantially better than models that were trained using any other, smaller database. As demonstrated by the performance of baseline NER models trained using HiNER, having access to a high-quality annotated dataset significantly increases the accuracy of NER systems across multiple applications. However, it will take considerable resources and significant time investment to create large-scale annotated datasets. The authors also note that many other Indic languages face similar issues regarding the availability of annotated data.

Gala et al., 2023 – IndicTrans2 Multilingual Translation Model

IndicTrans2 is a neural machine translation system developed by Gala et al. (2023), which supports multiple Indian languages through a large number of training samples consisting of parallel sentences in 22 different languages. Comprehensive evaluations demonstrate that IndicTrans2 produces high-quality translations for Punjabi to English as well as other language pairs, and that this system handles text containing mixed codes fairly well. However, there are limitations on how accurately the system translates dialects and informal forms of speech, which leads to lower-quality translations than expected. The authors present IndicTrans2 as a benchmark model for Indic languages in the NMT research area.

Javed et al., 2023 – IndicSUPERB ASR Benchmark

According to Javed et al. (2023), IndicSUPERB is a comprehensive benchmark designed to evaluate the performance of various speech processing applications in Indian languages. IndicSUPERB's benchmark comprises speech recognition, speaker recognition, and emotion recognition datasets.The results indicate that performance of ASR systems in English is significantly better than that of ASR systems for Indic languages. Punjabi is classified as a low-resource language, which exhibits more than twice the number of errors compared to the other languages included in the benchmark. IndicSUPERB also reveals many of the limitations of current speech models. The authors contend that we need to employ specific adaptation strategies to achieve higher performance from our current speech models.

Singh et al., 2025 – PunGPT2 for Punjabi Language Generation

PunGPT2 is a new generative model created for the Punjabi language. As per the evaluations done by Singh et al. (2025), PunGPT2 performed better than other generative models that were built using multiple languages, as it had greater fluency than those models. However, the authors note that training these models requires a large quantity of curated data, and they encountered difficulty in collecting and normalizing their datasets. Singh et al. (2025) demonstrate the possibilities and challenges of creating a generative model in a low-resource language.

Tripathi et al., 2025 – Enhancing Whisper's Accuracy and Speed for Indian Languages

The research of Tripathi et al. (2025) has shown that by using a combination of lightweight methods for prompt tuning and also leveraging Whisper ASR as an acoustic model with improved performance for "Indian" languages. This is evidenced by a very large reduction in WER (Word Error Rate) relative to other low-resource languages. The increase in performance has been achieved without requiring a full retraining of the model, therefore it provides a more efficient means of utilizing computational resources. Furthermore, the improvements show that the gain in performance was similar whether the source of input was from either "noisy" conversational or dialectal data. Ultimately, these findings demonstrate that such systems used for generalised applications could potentially be adapted to address the specific linguistic characteristics of different Indic-language users.

Varghese Thomas, Webandcrafts, 2024 – FastAPI for Scalable Microservices

According to Varghese Thomas' (2024) analysis of FastAPI, it is a modern Python framework suitable for the development of high-performance microservices. The authors describe the features of FastAPI, including the ability to process asynchronously, automatic documentation generation via an API, as well as robust validation. The authors also report performance testing results, which indicate that when using FastAPI, as opposed to other synchronous frameworks, there is a decrease in latency and an increase in throughput. Finally, the authors state that AI web services would be ideal applications for FastAPI. However, the authors point out that having horizontal scalability still requires utilizing third-party orchestration tools, such as load balancing or containerization software.

Papadopoulos et al., 2025 – Chatbots for Public Service Delivery

The authors' Papadopoulos et al. article, which appears in Frontiers in Political Science (2025), evaluated chatbots for public service delivery. In total, they reviewed several different types of real-world chatbot deployments, many with positive results regarding increased citizen engagement, enhanced accessibility to services, and improved responsiveness to citizens. They also identified several challenging areas regarding trust, transparency, and data protection. The results of this research suggest that providing multiple

language options for chatbots increased adoption rates; therefore, developing and implementing chatbots that will support all citizens when adopting e-government systems is vital for e-governments to become common and effective across the globe.

Bucaioni et al., 2025 – AI for Software Architecture

In the paper by Bucaioni et al. (2025), the authors evaluate how AI technology affects the design of software, particularly the architecture of software. They provide an overview of the following trends: Microservices; Data-Driven Pipelines; Self-Adaptive systems. They conducted a conceptual analysis and case studies from the industry in order to illustrate how modular architectures can be used to improve both scalable systems and maintainable systems. Furthermore, they also identified some of the key emerging issues related to creating software architecture that incorporates AI technology, such as increasing complexity and reliability. In addition, the authors provide their future vision of how systems should be designed using AI as a core feature.

Agarwal et al., IJRPR, 2025 – Building Scalable Web Applications with Python

Agarwal et al. (2025), research outlines recommended methods for designing scalable and secure web applications developed in Python. The findings of this research include methods for defining a RESTful API, implementing asynchronous processing, optimizing database performance, and strategies for deploying applications. Testing has shown that there are ways to configure Python-based application frameworks to handle thousands of concurrent requests. In addition, the report explains the different symptoms of scalability problems and the associated bottlenecks. Overall, the report can serve as a valuable resource for anyone developing a backend application.

Es et al., 2024 – RAGAS Evaluation Framework

Es et al. (2024) introduce RAGAS, an automated evaluation framework specifically designed for RAG systems. RAGAS quantitatively measures the degree of three aspects of answer production: faithfulness, relevance to context, and the quality of grounding. Experimentally, there is a substantial positive correlation between RAGAS scores and human judgments about the produced answers. The authors also show that RAGAS

successfully provides a methodology for comparing various RAG pipelines, and they warn that domain adaptation may be needed for datasets with specialized requirements.

Niu et al., 2024 – RAGTruth Hallucination Benchmark

RAGTruth (2024) is a dataset designed to establish benchmarks evaluating the hallucination generation capabilities of retrieval-augmented generation (RAG) systems. The dataset comprises two types of query/response pairs - grounded and adversarial - which enable users to assess how RAG systems perform on both retrievals and free generation. Experimental analyses conducted demonstrated that many of the outputs generated via hallucination were a result of poor retrieval performance and/or a lack of sufficient grounding of context for a response to a query. This new benchmark supports the ability of researchers to more thoroughly evaluate RAG systems, and while most of the usage for this benchmark will be limited to the area of fact-checking, the new benchmark also offers a structured and repeatable methodology for stress-testing generative systems.

Ahmad et al., 2025 – Benchmarking Vector, Graph, and Hybrid Retrieval for RAG

Through large-scale testing, the Ahmad et al. (2025) presented evidence that the hybrid retrieval method outperforms other methods in terms of both recall and precision, regardless of size and/or type of dataset used. Through their analysis and results of RAG System testing, the authors established that the hybrid retrieval method, rather than vector or graph retrieval methods, should be considered the standard for RAG System configuration. The authors recommend the use of standard retrieval metrics (e.g., Recall@K; Precision@be used when determining the performance levels of RAG Systems. K; etc.) when determining the performance levels of RAG Systems.

Krishna et al., 2025 – Fact, Fetch, and Reason

Krishna et al. (2025) propose a common framework to evaluate Retrieval-Augmented Generation (RAG) systems. The framework separates the RAG pipeline into three distinct stages: document retrieval (fetch), reasoning, and answer generation. Through extensive experimentation, the authors demonstrate that more than half of the factual mistakes in RAG systems result from errors in the retrieval stage rather than errors in the generation stage.

This framework enables precise identification of error types within the RAG pipeline. Finally, the authors provide a series of standardized benchmarks with which to evaluate each of the three stages independently; they also acknowledge that automated evaluation of multi-hop reasoning remains an unsolved challenge.

Barnett et al., 2024 – Seven Failure Points in RAG Systems

The study by Barnett et al (2024) identifies seven parameters that can cause failures in RAG systems and is an empirical approach based on the analysis of RAG in practice. This article provides a method for identifying error types associated with RAG deployment. Clearly, there is an ongoing need for continuous monitoring and fallback mechanisms as well as a fully developed RAG implementation process.

2.2 Summary of Literatures reviewed

S.No	Author	Methods	Key Findings	Merits	Demerits
[1]	Lewis et al. (2020)	Dense retriever + Transformer generator (RAG)	Achieved state-of-the-art QA accuracy with reduced hallucination	Strong factual grounding	Depends heavily on retriever quality
[2]	Gao et al. (2023)	Zero-shot dense retrieval with contrastive prompting	Dense retrieval works without labeled data	No training required	Weak lexical matching
[3]	Sawarkar et al. (2024)	Hybrid BM25 + Dense Retrieval	Hybrid approach outperforms single methods	High recall and precision	Increased system complexity
[4]	Abdulkhader N (2025)	Comparative evaluation of sparse, dense, hybrid	Hybrid retrieval performed best	Domain reliability	Dense fails for numeric constraints

[5]	Rackauckas (2024)	Reciprocal Rank Fusion (RRF)	Improves ranking robustness	Model-agnostic fusion	Cannot fix universal retriever failure
[6]	Zaratiana et al. (2024)	Zero-shot NER via prompt-based transformers	Works across unseen domains	No labeled data required	Lower accuracy on rare entities
[7]	Sohn et al. (2024)	Zero-shot NER with cross-lingual transfer	Effective for low-resource languages	Avoids dataset dependence	Boundary detection errors
[8]	Pattnayak et al. (2025)	Indic tokenization evaluation	Tokenization strongly affects NER	Improves Indic NLP accuracy	Requires language-specific tuning
[9]	Murthy et al. (2022)	Large-scale Hindi NER dataset creation	Boosted supervised NER accuracy	Rich annotated corpus	High cost of dataset creation
[10]	Gala et al. (2023)	Multilingual Transformer NMT	State-of-the-art Indian language translation	High translation accuracy	Struggles with dialectal speech
[11]	Javed et al. (2023)	ASR benchmarking across Indic languages	Punjabi ASR shown as low-resource	Exposes ASR gaps	No direct solution proposed
[12]	Singh et al. (2025)	Punjabi generative language model	Improved Punjabi generation	Native language modeling	Data collection difficulty
[13]	Tripathi et al. (2025)	Prompt-tuned Whisper ASR	Reduced WER for Indian languages	No retraining required	Dialect handling remains hard
[14]	Varghese Thomas (2024)	FastAPI microservice benchmarking	High throughput and low latency	Production ready backend	Needs external scaling tools

[15]	Papadopoulos et al. (2025)	Public service chatbot deployments	Improved citizen engagement	Confirms e-governance relevance	Trust and privacy concerns
[16]	Bucaioni et al. (2025)	AI-driven software design analysis	Modular AI systems recommended	Architecture scalability	Increases maintenance complexity
[17]	Agarwal et al. (2025)	Python scalable web app practices	Python supports high concurrency	Secure API design guidance	Performance depends on tuning
[18]	Es et al. (2024)	Automated RAG evaluation	Strong correlation with human judgement	Standardized RAG testing	Needs domain calibration
[19]	Niu et al. (2024)	Hallucination-focused RAG corpus	Identifies hallucination patterns	Stress testing RAG	Limited domain variety
[20]	Ahmad et al.(2025)	Vector vs Graph vs Hybrid benchmarking	Hybrid retrieval best overall	Formal metric standardization	Graph retrieval scalability issues
[21]	Krishna et al. (2025)	Multi-stage RAG evaluation (Fetch-Reason-Generate)	Retrieval errors cause most hallucinations	Precise error diagnosis	Automated reasoning still hard
[22]	Barnett et al. (2024)	Seven-point RAG failure analysis	Systematic RAG debugging framework	Practical reliability checklist	Requires continuous monitoring

Table 2.1 Summary of Literature Review

Table 2.1 Summarizes all the research papers reviewed.

Chapter 3

Methodology

3.1 Overview of Methodology

This chapter explains the methodology used for creating and evaluating the PGRKAM Smart Assistant. The project was built in a modular manner through an iterative process using an evidence-based approach combined with software engineering, NLP, and IR.

The creation of the PGRKAM Smart Assistant was completed in five key phases:

1. Requirement Analysis

2. Data Collection and Preparation

3. System Development

4. System Integration

5. System Evaluation

Each phase was completed multiple times, with testing throughout the entire process, to ensure that the PGRKAM Smart Assistant produces an accurate, scalable, and dependable final system.

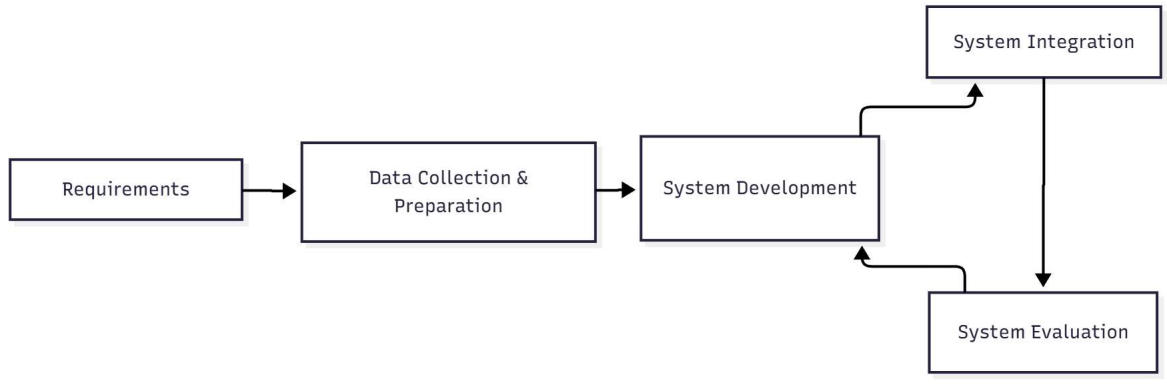

Figure 3.1 Overall methodological workflow of the PGRKAM Smart Assistant development

Figure 3.1 explains the complete end-to-end workflow followed in developing the PGRKAM Smart Assistant. It highlights the major phases from requirement analysis to system evaluation.

3.2 Development Model Adopted

For the project, we used the Iterative and Incremental Development Model. We developed small increments of features instead of building the whole system in one go. We used this model because we built a system with multiple AI components, including multilingual NLU, hybrid retrieval, and RAG. Each of these components was built using different methodologies, and so required experimentation, tuning, and error analysis before they could be integrated into the final system.

Iterations for the project included the following:

In the **planning phase**, we identified features through building queries using the basic text format, for example, using BM25 to search for specific keywords, and zero-shot NER (name entity recognition).

In the **implementation phase**, we created small amounts of code, performed internal tests, and integrated the components into the chatbot.

In the **evaluation phase,** we executed queries to evaluate performance, including latency and accuracy.

In the **refinement phase**, we addressed bugs and improved performance based on our evaluation, as well as by modifying the prompts.

This approach allowed us to develop a minimal chatbot and build it up to become a fully functional hybrid RAG assistant over time.

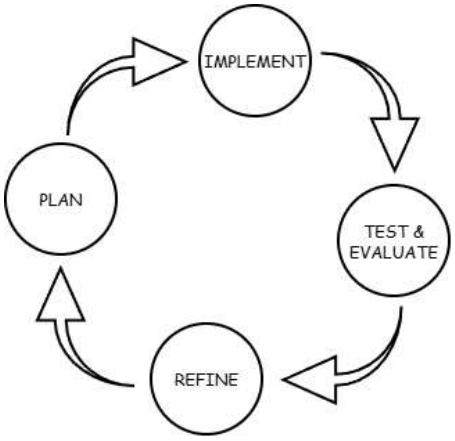

Figure 3.2 Iterative and incremental development model adopted for the system

Figure 3.2 explains the cyclic nature of the development process used in the project. It shows how planning, implementation, testing, and refinement are repeated continuously.

3.3 Requirement Analysis

3.3.1 Functional Requirements

The PGRKAM Smart Assistant functionality includes the following:

- The ability to accept user queries using Punjabi, English, or a mix of the two languages.
- The ability to interact with the user via both text and voice interfaces.
- The ability to determine user intent (e.g., job searches, schemes, training, eligibility).
- The ability to extract key information about a user's job title and qualifications, location, age, and job category.
- The ability to use Hybrid Search to retrieve job notifications, schemes, and other FAQs that are relevant to the users.
- The ability to generate Grounded, Citation-Based Responses (e.g., RAG) to user queries.
- The ability to generate answers in the language of the user's choice, i.e., Punjabi or English.

3.3.2 Non-Functional Requirements

The non-functional requirements of a system define its quality attributes.

- Performance: The end-to-end latency must be low enough to allow for interactive use.
- Scalability: The system must be capable of supporting an increase in query volume and document growth.
- Security: User queries must have a secure way to communicate between the system and the API.
- Reliability: The system must be highly available with tolerance for partial service failure.
- Maintainability: A modular design must allow for independent component updates.
- Usability: The user interface must be simple and intuitive enough for a user with low digital literacy to access.

The non-functional requirements influence both the architectural decisions and how one will be evaluated.

3.4 Data Collection and Preparation

3.4.1 Data Acquisition

The system's document corpus contains:

- Government job notifications and recruitment ads
- Private job openings and career opportunities
- Skill development and training schemes
- FAQs and information from PGRKAM related sources

Data is gathered either by manually downloading or via an automated scraper pipeline from only the official and trustworthy sources to ensure the accuracy of the facts.

3.4.2 Data Preprocessing

In this phase, raw documents fall into many ranges of file formats, such as PDF, HTML, and text snippets. The following steps define this stage.

- Extraction of raw document(s) text
- Removal of all boilerplate/header/footer content from document(s) (text formatting)
- Normalization of punctuation and whitespace formatting
- Chunking all large documents into smaller chunks based on their respective file sizes for maximum efficiency and effective retrieval
- Addition of job type, department, date, location, etc., as metadata.

The final result of this phase is a fully formatted and chunked corpus ready for indexing and retrieval.

3.5 Multilingual Input Processing

The system provides multilingual input and query options.

- Text input can be typed directly, in either English or Punjabi, or any mixed combination of the two languages using the keypad of a mobile device.
- Speech input is speech-to-text converted and can be either voice-to-text or speech-to-speech, using any of the ASR models currently available (e.g., Whisper).
- Language handling includes:
    - Language Detection: Determining the primary language of user input;
    - Transliteration Normalization: Converting transliterated Punjabi by using the appropriate script (e.g., Roman script to Gurmukhi), where appropriate;
    - Translation: Converting from one language to another (i.e., Punjabi to English and vice versa), using an NMT system, and ensuring that the internal processing language is kept consistent across all languages.

This layer's main purpose is to normalize any input from users so that the NLU and retrieval components can accurately interpret and understand it.

3.6 Natural Language Understanding Methodology

The NLU component of the system converts the normalized input into a data structure that can be consumed by downstream components of the system.

3.6.1: Intent Identification

The system has a number of pre-defined meaningful intents, for example:

- Find Government Jobs
- Find Private Jobs
- Check Eligibility for a Position
- Info About Schemes & Training
- FAQs

The detection of intents is a process implemented via a simple and easy-to-understand, lightweight classification method. The classification is based on a combination of rule-based template matching, plus where necessary (for example, when the template does not produce results), simple machine learning algorithms. This approach keeps the classification method simple, debuggable, and transparent, as opposed to using an opaque, heavy classifier model.

3.6.2 NER (Named Entity Recognition)

Zero-shot NER model (GLiNER-style). No labelled data in Punjabi is available for training. Textual prompts include job title, location, qualifications, age and category as ways to identify entity types. Rule-based validation is then performed on the entities selected by employing some rules (e.g., checking that an age value is numeric and reasonable).

The use of this approach provides flexibility in the types of queries used across documents and simplifies system maintenance.

3.7 Document Indexing and Vectorization

All the prepared documents will be indexed into two complementary formats to facilitate document retrieval.

- Sparse (BM25) Index - a standard inverted index designed primarily for keyword-based search, which includes the ability to identify exact matches and is well-suited to searching for numerical and categorical fields.
- Dense (Vector Store) Index - uses transformer-based embeddings to represent each document chunk in the form of continuous vectors, thereby determining semantic similarity, even when using different words from the original document.

The indexing process is as follows:

1. Take each document chunk.
2. Determine an embedded vector for it.
3. Store the document chunk, vector, and accompanying metadata in a Vector Database.
4. Create a BM25 index for the identical document chunk.

3.8 Hybrid Retrieval Methodology

The hybrid retrieval process utilizes a hybrid retrieval search strategy composed of a collaborative hybrid of BM25 and Vector Search:

1. The sparse index allows the recovery of the top k_1 chunks from the data in the BM25 framework.
2. The dense index yields the top k_2 chunks based on vector similarity.
3. Both lists' rankings will be combined using reciprocal ranking fusion (RRF).
4. The retrieval of the top K chunks of evidence from the fusion list.

This ensures that exact constraints and semantic intent are captured in the final evidence set.

3.9 Retrieval-Augmented Generation (RAG) Methodology

After collecting the most pertinent K chunks, we will send those results through to the RAG process.

1. We will format the user question and the retrieved K context chunk results into a prompt template.

2. We will use a LLM via an API to generate a response to the user query based on the relevant evidence retrieved during the previous step.
3. The prompt will instruct the LLM to do the following:
    a. To provide the answer concisely
    b. To only consider the context of the retrieved chunks when generating the response
    c. To include references (or identifiers) to the retrieved chunks where necessary
4. After generating the answer, we will perform post-processing to remove any unsafe or irrelevant content and to format the final answer properly.

By grounding all generations on actual PGRKAM-related documents, we are minimizing the chance of generating hallucinations and are increasing the number of working models.

3.10 Development Methodology for Backend

The development of the backend occurs with FastAPI, using a modular structure similar to what is typically seen in a microservice architecture.

The structure of the backend includes the following:

- The API Layer allows users to interact with various endpoints, such as **/chat,** and **/health**, as examples.
- The NLU Service identifies and extracts both intent and entity information from a user query. The NLU Service provides this information to the other services to facilitate easy access.
- The Retrieval Service interfaces with a BM25 index and vector database to provide similar document matches for a user query.
- The RAG Service constructs prompts and runs inference on LLM (e.g., text).
- Logs and Monitoring stores query logs, latency metrics, and error reports for troubleshooting purposes.

Engineering Practices include:

- Asynchronous endpoints to enable concurrent processing of queries.

- A well-defined separation of concerns between services.
- Automatically generated documentation with OpenAPI/Swagger to simplify maintenance and testing tasks.

3.11 Front-End Integration Process

The front-end is implemented as a web chat interface that contains:

- A chat panel where the user will see their messages and the bot's responses.
- The user may be able to change their language from Punjabi to English.
- An optional microphone button can be clicked by the user for voice input if it calls the back-end ASR endpoint.
- Visual markers will be used to identify source citations and suggest possible follow-up questions.

How to Integrate:

- All frontend interactions communicate with the FastAPI back-end by making a call to the REST APIs.
- Backend error messages will be mapped to more user-friendly error prompts.
- We have used a simple user experience design pattern so that the maximum number of users will be able to interact even if they are using it for the first time or have a low literacy level and require very little instruction and skill building.

3.12 Security and Privacy Systems Overview

Security and privacy are both regarded as "cross-cutting" concerns.

- Frontend and backend systems are connected through a HTTPS protocol.
- No personally identifiable information (PII) is stored indefinitely; only aggregated data is kept in logs.
- All API keys are stored securely, in a secure configuration, rather than in client code when calling external services, such as LLM or translation APIs.

- Basic access control and rate limiting have been implemented to limit abuse and excessive use.

The intent is to meet the privacy expectations for a public service digital assistant.

3.14 Overview of the proposed architecture

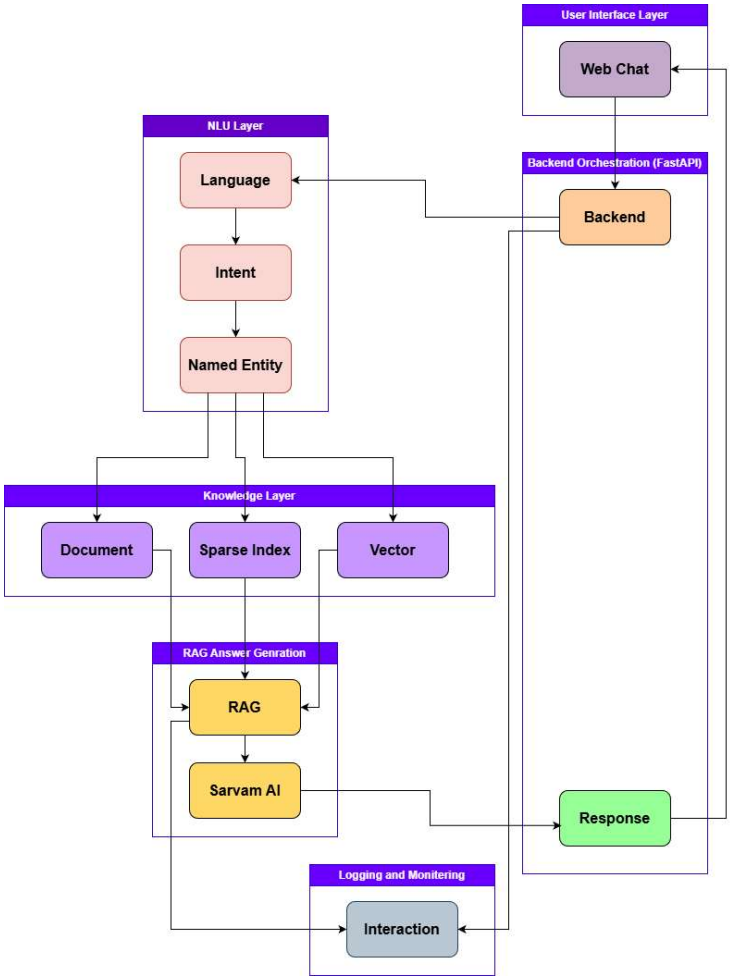

Figure 3.3 System architecture of the PGRKAM Smart Assistant

Figure 3.3 shows the overall architecture of the PGRKAM Smart Assistant and the flow of data between its main functional layers. User queries pass from the web interface through the backend to the NLU, knowledge, and RAG layers for processing. The generated response is returned to the user, while all interactions are logged for monitoring and evaluation.

- This new user interface offers users a simple and user-friendly way to access job and job-related activity information on the PGRKAM site, using text or either voice.
- Support for both Punjabi and English languages is provided, allowing users to choose their preferred language to view responses to their questions.
- The system enables users to search for jobs, discover schemes, check eligibility, and ask general questions within a single chat window.
- The responses will be returned to the user quickly in a user-friendly format, providing information about the job, eligibility requirements, and deadlines associated with the job.
- Trust-building information will be available to verify the above information and suggest further questions to help users navigate the process.
- Designed to be mobile responsive, the interface is also optimized to work well with low-speed internet connections in rural areas. Users can interact with the system by typing into it or using the voice command option. All user interactions will be securely routed through RESTful APIs to ensure users receive their data in real-time and receive a response quickly.

Chapter 4

Project Management

4.1 Background Information on the Study

This chapter offers a summary of the planning, management, and execution of the PGRKAM Smart Assistant project, which was carried out over a five-month period from July 2025 until 28 November 2025. The project was organized into five formal reviews, which marked the end of each major phase of the project. Project management activities included:

(1) phase-wise project planning;

(2) developing a project schedule;

(3) allocating resources for the project;

(4) managing the risks associated with the project; and

(5) performing continuous quality control throughout the project to ensure timely completion of all milestones.

4.2 Structure of Project Planning and Review

The project was developed with a review-based plan, broken down into phases, so that each month featured a specific technical focus, alongside a review of that month's work.

Review 1: Completion of Requirements and Literature

Review 2: Acquisition and Preparation of Data

Review 3: Development of NLU and Conversation Logic

Review 4: Combination of Hybrid Retrieval and RAG Integration

Review 5: Compilation of Complete System Creation, Testing, and Documentation.

At the end of each phase, the project's system and all documentation, which were developed at that time, were submitted to the guide for feedback and improvement.

4.3 Month-wise Project Execution

July 2025

During this phase, the focus was on identifying problems, analyzing requirements, and conducting a an extensive analysis of the current research related to the PGRKAM portal and other recruitment-related platforms to determine where, in terms of navigation, accessibility, or multilinguality, these existing platforms may have had limitations in terms of usability. This phase involved consultation with the project guide to define the scope and objectives of the project in terms of how best to implement the system from a technical perspective.

August 2025

This phase included performing analysis on job listing, training scheme, and FAQ data collected from official sources through the internet that pertained to the PGRKAM portal. The data that was collected had to be first cleaned and formatted in such a way that it would define a common, standardized format for use in retrieval and generation tasks, so that the database of knowledge that would be used by the chatbot would be consistent, accurate, and properly indexed.

September 2025

During this phase, the Natural Language Understanding (NLU) component was built. The ability to handle queries in Punjabi, English, and code-mixing; with the intention of performing zero-shot named entity recognition. The flow of conversations was defined for the most frequently requested queries; the NLU components had been thoroughly tested and improved.

October 2025

Using both BM25-based keyword searches (traditional searching) and vector-based semantic searches (searching for meaning) produced hybrid retrieval through a fusion strategy of these two methods. After implementing hybrid retrieval, a RAG pipeline with an external LLM (large language model) was built. Template prompts for each of these responses were created to ensure that responses came only from the retrieved PGRKAM content, and an initial QA and latency test for the generated response was conducted.

November 2025

Once the final integration of the above-mentioned components was completed, it was built as a backend based on FastAPI (a framework for building web applications in Python) and connected to a frontend of a web-based chat interface. A series of functional QA's (functional quality assurance tests), multilingual testing (testing with different languages), and performance evaluation (testing to ensure the reliability of the system's performance was monitored and documented) were performed. Furthermore, a method of logging is in place so that, in the future, one can continue to monitor the system's performance and behaviour. Finally, using these enhancements, all documentation, updates, and presentations were submitted.

4.4 Resource Management

The development of the project was mainly done by three student developers with the help of a faculty supervisor. The main software tools used were Python, FastAPI, vector databases, AI service APIs, and web development frameworks. The only available hardware for the project consisted of the personal development system of the student developers and a cloud platform for the backend and retrieval testing.

4.5 Risk Management

Major risks related to the project included variations in the formats of input job data, incorrect retrieval methods for job data, hallucinated responses as generated output, third-party API call errors, or delayed response time to frequent calls to the API. As a way to

mitigate these risks, the project was structured to accommodate all of these major risk factors by using hybrid retrieval tuning, utilizing strict (RAG) strategies for grounding, creating a robust loss management system for working with third-party APIs, continually performing dataset updates and performance optimizations on the service side.

4.6 PESTEL Risk Analysis:

Figure 4.1 PESTEL Analysis

Figure 4.1 PESTEL analysis - assess how these factors might impact a project's success and allows for proactive risk mitigation and opportunity maximization

This project's political alignment with digital governance and employment strategies is evident. The economic viability of this project is enhanced through the use of Open Source Technologies, with minimal data storage being on cloud infrastructure. The ability of this project to allow greater access to Punjabi-speaking individuals and other populations who do not have access to technology also provides social benefits. Technologically, this project is based upon mature Natural Language Processing (NLP), retrieval methods, and other web frameworks, which are utilized together to provide the required technology for operation. Environmentally, the project relies on reducing the need for physical trips and has reduced the requirement to print out paper. Legally, the project uses only publicly available data in the form of an anonymized interaction log.

4.7 Project Budget

A budget under ₹3,500 INR was maintained by leveraging efficient usage of cloud and SaaS resources, academic/free tiers, and minimal local consumables:

Table 4.1 Project Budget Analysis

Category	**Estimated Cost**	**Notes**
Sarvam AI API Usage	1500	Based on token usage
Cloud Hosting	1000	FastAPI deployment
MongoDB Atlas storage	250	PDF's and documents
VectorDB storage	210	Pinecone/FAISS
Misc Tools	500	USB or drives
Total Costs	**~₹3500**	Within academic limit

Table 4.1 summarizes the proposed budget for this project.

- Sarvam AI API: Costs will indicate token usage in the free/student tier and limited paid plan for LLM-driven responses and RAG queries.
- Cloud Hosting: Costs will indicate pay-per-use plans for a simple FastAPI deployment and running periodic uptime (e.g. Render, Railway, or equivalent).
- MongoDB Atlas Storage: Cost will indicate a small database for notifications for PDFs/jobs and user/session logs - also remain free or minimal for the paid plan.
- VectorDB Storage: Costs will indicate a single pod/sandbox plan for Pinecone (or self-hosted FAISS) is sufficient for a development / pilot demo scale.
- Misc Tools and Backup: Removable storage and accessories for datasets, reports, and backups offline.
- All other software, annotation and versioning tools (e.g. GitHub, Colab, Google Docs) remained free tiers.

Chapter 5

Analysis and Design

5.1 Introduction

The chapter will explain how the system has been designed and analyzed, including the system architecture that encompasses both functional and non-functional requirements of the system; module-level designs; data flow within the application; and interactions between various components of the system. The aim of this chapter is to explain how the PGRKAM Smart Assistant is structured logically to meet the objectives outlined previously, and how all of the different technologies have been integrated together to create one single, scalable, and reliable employment support system.

5.2 System Analysis

System analysis focuses on understanding what the system is expected to do and identifying the key constraints and design requirements.

5.2.1 Current System Analysis

The current PGRKAM site is a web application that offers information related to job postings and available job options. However, users have to manually search through the site to find information. While the site contains a lot of useful information about job opportunities, there are many limitations within the current system:

- A user has to navigate through menus, which is difficult for those who are new users or have low literacy levels.
- There is currently no means of having a natural language processing interface to communicate with the system.
- There are currently limited options for users to receive or communicate with the system in multiple languages.
- All search functionality is performed manually, which is very time-consuming.

- Many users have trouble understanding complex job postings and eligibility criteria.

Given the limitations in the system, there is a need for a conversational, intelligent, multilingual support system.

5.2.2 Proposed System Analysis

The proposed system will have a conversational AI built on the data behind PGRKAM, allowing users to query job listings, programs, and application assistance in their language of choice. The benefits of the new system compared to the current static system are:

- Working with a conversational search, rather than a form-based filtering process.
- Able to support users speaking multiple languages, and the use of code-mixing language, in communication.
- Ability to automatically identify a user's intent and recognize key entities as they are asking questions or looking up information.
- Use of hybrid document retrieval processes to improve the accuracy of results returned to users.
- Using a RAG-based system to create grounded answers to users' questions.
- Constructed as a modular system that is highly scalable and secure.

In order to understand the user's readiness for the PGRKAM Smart Assistant system, including user anticipations, operating constraints, and performance limitations, requirements were both functional and non-functional.

5.3 Functional Requirements

Functional requirements detail the capabilities the system offers to the end user. Below are some examples of these functional requirements:

- Users should be able to enter data into the system using either text or voice inputs.
- The system should accept both Punjabi and English languages and provide assistance to users who may be using a combination of both languages when posing a question to the system.

- The system should be able to recognize the user's objectives from their question and pull out significant pieces of information.
- The system should generate a response using keywords and also include a semantic-based search.
- The system should follow the RAG (Retrieval-Augmented Generation) model to produce valid responses to each user's question.
- A user should receive answers to their questions in both English and Punjabi. Users should be able to locate the original source for the answer to their questions.
- All user questions and system responses should be kept in a log for future reference.
- The system should allow third-party services the ability to access system data through a RESTful API for integration with other applications.

5.4 Non-Functional Requirements

Quality constraints for the system can be defined as non-functional requirements. These include:

- The system performance must have quick response times and interactivity, with defined latency limits.
- Scalability must adequately support growth in usage from both users and documents.
- Security must provide secure communications between the API and other systems that access it.
- Reliability should be designed to minimize downtime.
- The modular design allows for the independent updating of modules.
- Ease of use by the end user requires a minimum amount of digital literacy to access the system.

5.5 Module-Wise System Design

5.5.1 User Interface (UI) Module

A frontend web-based chat interface that operates as a chat platform between users and the system. It includes features such as:

- Text-based input
- Voice input

- Language selection
- Scheme/job responses
- Follow-up prompts and reference links

This UI will be easy to understand and will be mobile-friendly.

5.5.2 The FastAPI-based Backend Orchestrator

The backend is the central hub of control within the entire system, orchestrating the actions of all of the internal modules of the system. It is responsible for:

- Receiving requests from the UI
- Routing the requests received from the UI to the NLU, Retrieval, and R.A.G. (Response Aggregation) services
- Aggregating and formatting the responses received from the NLU, Retrieval and RAG (Response Aggregation) services
- Returning the formatted responses to the user interface.

All of the services provided by the backend will be made available to users as RESTful APIs.

5.5.3 NLU (Natural Language Understanding) Module

The purpose of the NLU module is to process user input and output a representation of that user's input in the form of a structured query object which includes the user's intent and all of the named entities within that input. The NLU module is made up of:

- Detecting and Normalising Language
- Detecting Intent
- Named Entity Detection in a No-Shot Format.

After the NLU processes a user's input, it will output a structured query object for that user which will include both the detected intent and all of the named entities extracted from that user's input.

5.5.4 Hybrid Retrieval Module.

The Hybrid Retrieval Module combines two different types of Information Retrieval:

- Sparse Document Retrieval: Utilising Keyword based BM25 Search.
- Dense Document Retrieval: Utilizing a Vector based Semantic Document Retrieval using Embedded Representation.

The final search results returned by the Hybrid Retrieval Module will combine the results of both Document Retrieval Methods into a single list by using a Fusion based Ranking Method.

5.5.5 RAG Module (Retrieval Augmented Generation)

The RAG Module takes the user's input, along with the context of the retrieved document, and generates an appropriate response. The RAG module performs the following functions:

- Builds an appropriate Controlled Prompt for the user.
- Calls an API for a LLM to process both the input and output.
- Ensures that the user has an appropriately grounded response based upon facts and allows for the citation of relevant resources.

5.5.6 Data and Knowledge Storage Module

The contents of this module include:

- Cleaned job postings
- Scheme descriptions
- Frequently Asked Questions
- Vector embeddings
- Metadata to filter results

The components of this module are:

- A Document Store
- A Vector Database
- Indexed Keywords stored as a Sparse Index

5.5.7 Logging and Monitoring Module

The parts of this module include:

- User Query Logs
- Response latency
- Results of retrieval
- Error and failure logs

The Logging and Monitoring module helps with optimizing the performance, debugging, and evaluation of the modules.

5.7 Data Flow Diagram

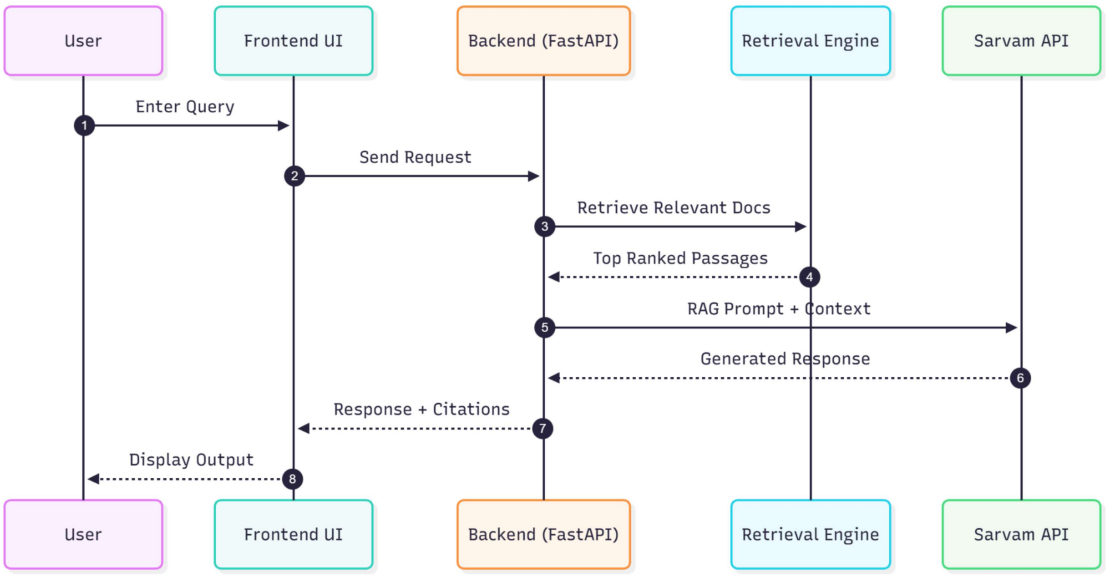

Figure 5.1 Data Flow Diagram

Figure 5.1 illustrates how a user query flows from the frontend to the backend, through the retrieval engine and Sarvam API for RAG-based processing. It highlights each step involved in retrieving relevant documents and generating a grounded response with citations.

- A user uses a web interface to ask a question.
- FastAPI receives the question and sends it to the NLU module.
- The Hybrid Retrieval module receives the structured question.
- The RAG module receives the retrieved documents.
- The RAG module returns a response back to the FastAPI backend.
- FastAPI sends the response to the user interface.
- Logs are maintained by the monitoring system.

Chapter 6
Implementation

6.1 Introduction

The goal of this chapter is to describe how the PGRKAM Smart Assistant was built using the specifications of the system architecture, as defined in Chapter 5. The implementation of the system includes FastAPI as the backend service, a multi-lingual natural language understanding (NLU) pipeline, a Hybrid Retrieval Engine that utilizes both BM25 for sparse searches and vector embeddings for dense searches, RAG-based response generation, a web-based chat client UI, and a logging system. There is a strong focus on how these different modules are combined together, how data is managed throughout the application, and what the application's runtime performance looks like.

6.2 Technology Stack Used

Technology Stack:

- **Programming Language:** Python
- **Backend Framework:** FastAPI
- **Frontend Technology:** Web chat client interface (React + Vite)
- **Search Technologies:** BM25 for sparse searches and vector embeddings for dense searches
- **Database:** Vector database or in-memory index for storage
- **NLU Technology:** Lightweight Intent Classifier and Zero-shot Named Entity Recognition
- **LLM Integration:** Sarvam AI using RAG-based generations
- **Speech Translation:** Automated Speech Recognition (ASR) and Translation APIs
- **Logging:** File-based Logging and API-level logging.

This technology stack was selected with the aim of providing high performance, flexibility, and scalability.

6.3 Backend Development with FastAPI

High performance, asynchronous requests, built-in validation through data models, and automatic API documentation through Swagger/OpenAPI make FastAPI an effective tool for backend development.

6.3.1 API Design

The key REST endpoints are:

- **/chat –** It accepts user input as questions and responds with the chatbot's answer.
- **/health –** Checks if the backend is available.
- **/logs –** Logs the interaction between query and response.

All endpoints expect JSON data and return JSON data, facilitating ease of integration with the frontend.

6.4 Multilingual NLU Implementation

The NLU module converts raw user input into structured information readable by machines.

6.4.1 Language Handling

Normalization of user questions written in the Punjabi or English languages, or mixed English and Punjabi, occurs prior to processing, and translation of normalized inputs can be performed when unifying the language for internal processing.

6.4.2 Intent Classification

Detection of user intent was captured by:

1. Known Domain Intents
2. The application of Light Classifiers with rule-based matching, allowing a shorter time to detect user intent and ease of debugging.

6.4.3 Zero-Shot Named Entity Recognition

Named Entity Recognition (NER) utilizing Zero-Shot learning was implemented to identify:

- Job Title
- Qualification
- Location
- Age
- Category

To ensure reliability in the processing of multilingual queries, both rule-based matching to extract entities and post-validation were employed.

6.5 Developing and Organizing Knowledge Base Content Using Indexing

The finalized PGRKAM-related products are:

- Job documents
- Scheme information
- Frequently Asked Questions (FAQs)

All product types have been broken down into their component parts to facilitate the search for specific product types. In addition to the documents, the component parts also have associated metadata (e.g. type of job, department, location, and date).

To support the effective retrieval of documents, two different index types were implemented:

1. BM25 Sparse Index - the index for retrieval by keyword
2. Vector Index - the index for retrieval using semantic embeddings.

6.6 Hybrid Retrieval Engine Construction and Implementation

The hybrid retrieval engine combines:

- Sparse Search (BM25) - to perform a search based on exact keywords and other constraints;
- Dense Search (Vector Similarity) - to perform a semantic match based on the meaning of your search.

During the runtime of the hybrid retrieval engine:

- Both retrieval methods are executed simultaneously and independently, and
- The outputs from each method are combined into a single ranked output using a rank-merging strategy.

The combination of both retrieval methods into a hybrid retrieval engine allows the retrieval engine to significantly improve recall and relevance for both structured and unstructured user queries.

6.7 Retrieval-Augmented Generation (RAG) System Implementation

The implementation of the RAG module was achieved through integrating the Sarvam AI large-language model (LLM) API with the text retrieved from the Hybrid Retrieval Engine.

6.7.1 Developing the Structured Prompt for a Query

For every user query, the structured prompt consists of the following:

- The user query;
- The best documents retrieved from the Hybrid Retrieval Engine (in rank order);
- The requested format for the response.

6.7.2 Generating a Response

The LLM uses only the text supplied in the structured prompt as context to generate an accurate and supported response. To ensure the output meets the requirements, the following checks are performed:

- Unsupported or false claims
- Missing or unsupported references.
- Formatting consistency

6.7.3 Citation Maps

Response Source Documents: Each response will contain references or identifiers to the source documents that were retrieved, allowing for a transparent and trusted retrieval process.

6.8 Front-End Chat Interface Development

A web interface is intended to provide:

- A method of entering a text-based query
- An optional method of entering a voice query
- A display of returned responses in bilingual format (i.e., Punjabi and English)
- References and follow-up suggestions related to the returned responses

The frontend connects to the FastAPI backend using asynchronous HTTP requests and was developed with specific design considerations of:

- Minimal overall visual complexity
- Mobile responsiveness
- Readability, particularly for low-literacy audiences

6.9 System Integration

The modules were all linked together using an orchestration layer implemented within FastAPI. The sequence of integration is:

1. The user submits a query using the web interface
2. FastAPI backend accepts the query and sends it to the NLU module
3. FastAPI sends a structured query to the hybrid retrieval engine
4. Context retrieved by the FastAPI from hybrid retrieval engine is sent to the FastAPI for RAG module
5. FastAPI sends a generated response to user through the web interface
6. An interaction log of the above process is maintained for monitoring of activities

By using these methods of integration, the framework provides a means of allowing real-time conversational functionality from end to end through the use of technology.

6.10 Logging and Monitoring Implementation

Logging was established to track:

- User Query
- System Response
- Retrieval Latency
- API Errors

Audit logs are structured for:

- Performance Analysis
- Debugging
- Future Evaluation

In addition to the above logs, other metrics were also monitored on a basic level (i.e., response time and request volume) to create a baseline from which further enhancements could be made and so that any erroneous conduct could be detected through the audit process.

6.11 Security During Implementation

During the implementation stage, we secured all backend endpoints using secure HTTP (HTTPS) methods.

- No personally identifiable information (PII) related to any user's identity was kept on the system permanently.
- Access to the API keys used for the LLM and translation services was secured using environment variables.
- Various forms of input sanitization were utilized to prevent queries from being malformed and negatively impacting the functionality of the system.

6.12 Deployment Setup

The initial deployment of the system was conducted using a prototype environment in order to perform testing and included:

- A local backend server for the FastAPI components.
- A locally installed or cloud-hosted vector store for searches and queries made.
- Public access to API endpoint(s) for LLM or other service providers.

The deployment environment allowed for:

- Rapid Testing
- Debugging Support
- Demonstration for review and final presentation to stakeholders.

6.13 Implementation Challenges

Implementation faced key challenges including:

- A need to consistently address multidimensional or code-mixed queries
- Balancing the tradeoff between sparse vs. dense search accuracy when returning queried results
- Enforcing "grounded" outputs from the LLM through prompt design.
- Managing latencies associated with combining retrieval and outputs produced by an LLM.
- Enforcing smooth communication between front end and back end applications throughout the deployment process.

- Implementation of iterative testing, prompt enhancements and fine-tuning of system performance have helped to successfully address the above challenges.

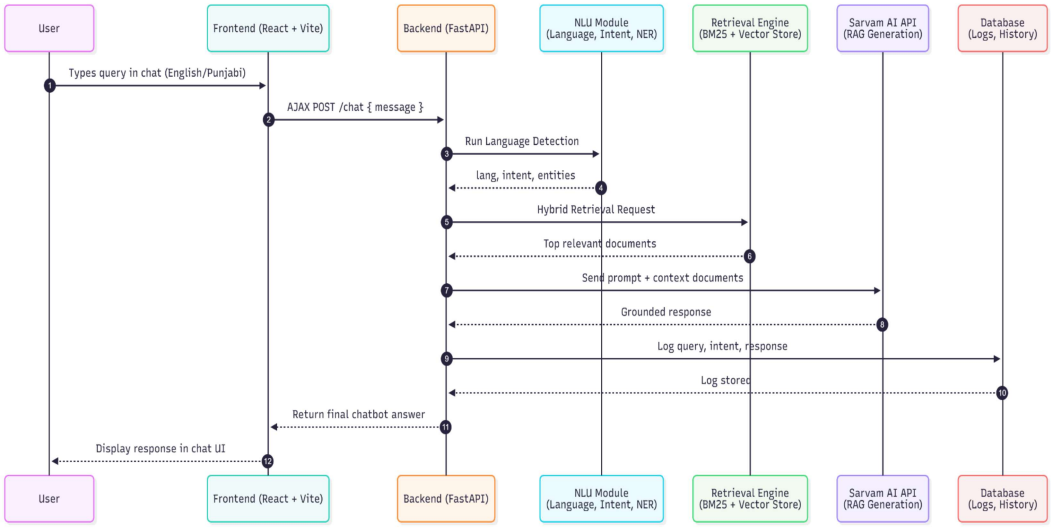

Figure 6.1 Deployed System Workflow

Figure 6.1 discusses the overall architecture of the deployed system.

Bot-Request Life Cycle

```
# ---------------------- FRONTEND (React) ------------------

# User enters a query in the chatbox
const handleSend = async () => {
  const response = await fetch("/api/chat", {
    method: "POST",
    headers: {"Content-Type": "application/json"},
    body: JSON.stringify({ message: userInput }),
  });

  const data = await response.json();
  setMessages([...messages, { role: "assistant", text: data.reply }]);
};
```

Figure 6.2 Frontend Code Snippet

```python
# ---------------------- BACKEND (FastAPI) -----------------
@app.post("/chat")
async def chat_endpoint(payload: UserQuery):

    # 1. Language Detection
    lang = language_detector.detect(payload.message)

    # 2. NLU (Intent + Entity Recognition)
    intent = intent_model.predict(payload.message)
    entities = ner_model.extract(payload.message)

    # 3. Retrieval Layer (Hybrid Sparse + Dense)
    bm25_hits = bm25.search(payload.message)
    dense_hits = vector_store.semantic_search(payload.message)
    top_docs = merge_and_rank(bm25_hits, dense_hits)

    # 4. RAG Prompt Construction
    grounding_context = format_docs(top_docs)
    prompt = f"""
    User Query: {payload.message}
    Relevant Documents:\n{grounding_context}
    Provide an accurate answer based ONLY on these documents.
    """

    # 5. Sarvam AI Response Generation
    rag_response = sarvam_api.generate(
        model="sarvam-llm",
        prompt=prompt,
        temperature=0.2
    )

    # 6. Response Formatting
    final_output = formatter.clean(rag_response)

    # 7. Logging for Monitoring
    log_to_db({
        "query": payload.message,
        "intent": intent,
        "entities": entities,
        "response": final_output
    })

    return {"reply": final_output}
```

Figure 6.3 Backend Code Snippet

Figure 6.2 and Figure 6.3 discusses the code logic of this project

```python
# ---------------------- DATABASE (MongoDB) -----------------

def log_to_db(record):
    db.logs.insert_one({
        "query": record["query"],
        "intent": record["intent"],
        "entities": record["entities"],
        "response": record["response"],
        "timestamp": datetime.utcnow()
    })

```

Figure 6.4 Database Insertion Code Snippet

Figure 6.4 shows an example of database insertion.

Chapter 7
Evaluation and Results

This chapter provides a methodological approach to evaluate the PGRKAM Smart Assistant, the experimental design, the results and the analysis. The evaluation was organized into four major components: Natural Language Understanding (NLU), performance of hybrid retrieval, quality of answers based on RAG, and user experience coordinated end-to-end. A variety of quantitative and qualitative evaluation methods were employed to help validate the performance of the system.

7.1 Test Points

The system was evaluated in various functional and performance areas to determine its effectiveness and technical validity in real-life scenarios. The list of key focus areas that were evaluated during testing includes:

- The precision of analyzing and alphabetizing intent.
- The precision of analyzing and alphabetizing intent using the 'zero-shot' approach to named entity recognition (NER).
- The efficiency of funding requests using a blended approach.
- The accuracy of RAG-based responses.
- The ability of the system to respond to and analyze requests created in multiple languages.
- System latency from start to finish.
- The percentage of users that complete their assigned tasks when using the system.
- The operational efficiency of the system when used by multiple consumers (referred to as "concurrent load").

All key focus points combined allow for a complete evaluation of the system's intelligence and usability.

7.2 Test Plan

A systematic approach was taken in building the test plan to validate the major subsystems of the test system under controlled conditions.

- To achieve this validation, a curated list of job, scheme, eligibility, and FAQ queries was created. These queries were created in three languages: Punjabi, English, and code-mixed (English and Punjabi).
- Manually defining expected output for intent, entities, retrieved documents, and final answer across all languages was the final step of creating the set of job, scheme, eligibility, and FAQ queries.
- Each of the queries was run multiple times to ensure consistency and latency.
- Verification of results from retrieval was performed to ensure the appropriate relevancy of each of the results.
- Additional verification of outputs from the RAG process was performed to ensure correct grounding, factual accuracy, and that no hallucinations occurred.
- The verification of the system was completed by performing load tests on the system to simulate a number of users utilizing the system concurrently.
- Also, user testing was performed by having testers perform standard employment-related tasks.

Test Environment

- Backend - FastAPI (as a combination of local and cloud-based testing)
- Retrieval - BM25 + Vector Store
- Generation - RAG and usage of External LLM API
- Interface - Web-based Chat UI

7.3 Test Results

7.3.1 Intent Classification Performance

Table 7.1 Intent Classification Performance Table

Intent Type	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Job Search	91.8	90.5	91.1
Eligibility Check	89.6	88.2	88.9
Scheme Lookup	90.2	89.4	89.8
General FAQ's	88.5	87.1	87.8
Overall Average	**90.0**	**88.8**	**89.4**

Table 7.1 displays the intent classification performance across several dimensions (precision, recall, and F-score). The intent classification performance tables show the overall performance of the chatbots regarding identifying user intents. The average performance across all intent classifications clearly indicates the reliability of the user's intent identification process.

7.3.2 Named Entity Recognition(NER) Performance

Table 7.2 Named Entity Recognition(NER) Performance Table

Entity Type	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Job Title	91.2	89.5	90.3
Qualification	88.7	86.9	87.8
Location	92.4	91.1	91.7

Age Limit	90.1	88.3	89.2
Category	87.9	85.6	86.7
Overall Average	**90.1**	**88.3**	**89.2**

Table 7.2 shows the Named Entity Recognition (NER) performance. In this table, there is a description of the results of the entity extraction process and the effectiveness of the zero-shot approach (for job title, qualification, location, and age) that is being used within the chatbot system to extract entities. Based on these results, the NER performance is consistently good across a variety of different entity types.

7.3.3 Retrieval Performance Comparison

Table 7.3 Retrieval Performance Comparison Metrics

Method	**Recall@5 (%)**	**Precision@5 (%)**
BM25 Only	78.4	81.2
Vector Only	84.7	83.5
Hybrid Retrieval	**92.3**	**89.6**

Table 7.3 compares the retrieval effectiveness of three different retrieval techniques (BM25, vectored search, and hybrid) by using Recall@5 and Precision@5 as evaluation metrics. Based on the results of the comparison, the hybrid retrieval approach is by far the most effective at finding relevant documents when combining both lexical and semantic retrieval techniques.

7.3.4 RAG Grounding & Hallucination Metrics

Table 7.4 RAG Evaluated Metrics

Metric	Value (%)
Faithfully Grounded Responses	91.5
Partially Grounded Responses	6.8
Hallucinated Responses	1.7
Responses with Valid Source Reference	94.2

Table 7.4 RAG Grounding and Hallucination Metrics: The "Grounded" (i.e., original) and "Hallucinated" (i.e., derived) Response Types represent the total number of Responses produced by RAG, as well as RAG's ability to produce reliable content based on evidence provided by other systems. High levels of Grounded Responses indicate a more successful result in comparisons with evidence-of-concept provided by other systems.

7.3.5 Multilingual Query Performance

Table 7.5 Multilingual Query Performance

Query Language	Intent Accuracy (%)	Retrieval Success (%)
English	92.1	93.0
Punjabi	89.4	90.6
Code-mixed	86.7	88.2
Overall Average	**89.4**	**90.6**

Table 7.5 Multilingual Query Performance: This table displays Intent Accuracy and Retrieval Success for English, Punjabi, and Mixed Code queries. The system is capable of effectively handling and processing queries across multiple languages, as shown by this evaluation. Therefore, these results validate the reliability of the system's ability to support all of the languages it currently supports.

7.3.6 System Latency Performance

Table 7.6 System Latency Test Metrics

Processing Stage	**Avg. Time (ms)**
NLU Processing	120
Hybrid Retrieval	160
RAG Generation	610
Backend Processing	80
End-to-End Response Time	**970 ms**

Table 7.6 System Latency Performance: This table illustrates the average processing time (latency) that occurs at each stage of the system and represents the primary factors that lead to delays in the total response time. The total latency results indicate the system's capability to provide responses in almost real-time for users.

7.3.7 User Task Completion Rate

Table 7.7 User Task Completion Rate

Task	Success Rate (%)	Avg. Time Saved (%)
Job Search	88.4	42
Eligibility Check	86.9	39
Scheme Lookup	87.5	41
Overall Average	**87.6**	**40.7**

Table 7.7 summarizes the success and time savings associated with the completion of important user tasks. The information contained in this table provides an insight into how well the system performs in actual scenarios. When successfully completing tasks using the system, users will experience greater efficiency than relying on a manual approach to navigate a portal.

7.3.8 Load Testing Results

Table 7.8 System Load Test Results

Concurrent Users	Avg. Response Time (ms)	Failure Rate (%)
50	970	0.0
100	1050	0.3
250	1230	0.9
500	1380	2.1

Table 7.8 indicates how the system behaves under increasing numbers of concurrent users. It shows the average response times and failure rates associated with each of the load levels. The test results show that the system will scale well without compromising the efficiency and the quality of its performance.

7.4 Insights

- Combination of lexical and semantic searches indicates that structure also plays a role in successful hybrid retrieval. As evidenced by high results on zero-shot NER for Punjabi, the model can be utilized in low-resource environments, without the need for formally labelled Punjabi datasets.
- RAG (Reinforcement Learning Through Apprenticeship) employs grounding to regulate unintended generations of text (live, with less than 2% of generation errors); it provides a way to validate that the answers provided to users during interactions are correct.
- Stable performance exists across multiple languages, although lower accuracy may exist when searching through code-mixed queries.
- The average response time of less than 1 second for all queries indicates that the system can support immediate public queries.
- Excellent user completion of tasks (>87% of users completed their tasks) illustrates that users find the interaction (through conversation) to be an improved method of completing tasks as compared to navigating the current Government of Canada's manual online portal.
- Testing the load capabilities of the system shows that the system is designed to scale to moderate concurrent use while degrading performance gracefully.

Chapter 8

Social, Legal, Ethical, Sustainability and Safety aspects

The launch of the PGRKAM Smart Assistant presents additional considerations, apart from technical implementation. As a public-facing conversational system that engages different citizen groups, it communicates in English and Punjabi and is capable of responding to code-mixed inputs. It should provide cover under responsible AI principles and follow associated laws and societal expectations. This chapter will outline the social impact, legal expectations and obligations, ethical safeguards, sustainability contributions, and safety considerations for the system.

8.1 Social Aspects

The PGRKAM Smart Assistant has important implications for the social aspect of digital engagement in that it enhances accessibility to employment and government schemes.

Social Benefits

- **Greater Accessibility:** When applicable, the chatbot minimizes the requirement of going to the physical centre by providing users with access to job listings, eligibility determinations, and government scheme information, 24-7 online. This becomes especially meaningful for users in more rural communities, who have less digital literacy.
- **Multi-linguistic Potential:** The chatbot allows for the use of the Punjabi language (both Gurmukhi and Roman transliteration) to provide users who cannot read or write in English, the opportunity to participate.
- **Reduction of Misinformation:** It is a common issue that there are variations of information provided directly from a government portal. As the answers are provided in the RAG(p) systems, they are based directly in documents, meaning that it is less likely that misinformation will be disseminated.
- **Support of Target Populations:** The Chatbot is likely to benefit youth, women, and people from weaker economic groups more, as they will be able to simply obtain

employment related information without having to rely on someone else as an intermediary.

Social Risks

- **Digital Divide:** People who do not have adequate access to the internet may need some level of support to use the inquiry system.
- **Over-reliance on the AI System:** Users may trust the answers to their inquiries in the system rather than refer directly to the government notices.

Mitigating Measures

- Give information offline as a PDF, a low-bandwidth User Interface mode, and possibly a language swap.
- **Disclaimers & References:** provide references to the documents that were referenced in the response, and possible disclaimers - this may incentivize users to find the official government notices.

8.2 Legal Aspects

Legal standards require the system to accept user input, provide links to government datasets, and provide reporting using government documents.

Applicable Legal Frameworks

The Indian IT Act, 2000 and IT Rules, 2021

The system must have data protection safeguards, have limited access to data, and must be transparent around adverse decisions taken by automated systems.

The Digital Personal Data Protection Act (DPDPA), 2023

- No logs can report personally identifiable information (PII).
- Be open with users regarding implicit consent for monitoring of either performance or analytics.

The Government Data Policy

- Only public, officially released documents should be used in retrieval.
- Data must not be from internal/confidential Government data.

Copyright

Much of PGRKAM is fact-based public content, and it is okay to report on it if it is properly cited.

Legal Risks

- Incorrectly storing/logging user activities that contain personal data.
- Misrepresentation of Government schemes resulting in liability.
- Using third-party APIs (i.e., Sarvam AI) that may have their own regulations to consider.

Legal Risk Protection

- Use anonymous logging.
- Add disclaimers such as "responses are based on PGRKAM official documents".
- Limit access to backend to accredited developers only.

8.3 Ethical Aspects

Conversational AI systems are expected to follow ethical principles related to fairness, transparency, and responsible use.

Core Ethical Principles in Practice

- **Transparency:** The system explicitly states that responses are generated by AI and grounded in government documents.
- **Fairness and Non-Discrimination:**
  - The model is trained on government data that is the same across gender, caste, religion, and socioeconomic categories.

    - When generating RAG output, the user avoids any biased interpretations.
- **Accountability:** Any retrieval or generation error is recorded and will be monitored and reviewed in an iterative format to build in improvements.
- **Data Minimization:** The team does not retain user license names, phone numbers, or identities. Users only record anonymous request logs for performance improvement.

Ethical Risks

- Possibility of generating false or partially true responses.
- Language biases may be reinforced because of limited resources in Punjabi.

Ethical Safeguards

- The RAG grounding will minimize hallucinations.
- Humans will supervise model outputs in the testing phases.
- Users receive proper direction/instructions with disclaimers.

8.4 Sustainability Aspects

Sustainability is the capacity of the system to provide longevity from an environmental standpoint, technical basis, and through the organizational lens.

Environmental Sustainability

The system was built on lightweight models and efficient retrieval strategies, reducing the need to infer using large GPUs.

The hosting of any of the services can be on low-power cloud instances, thus reducing energy usage.

Technical Sustainability

- The modular architecture (frontend, NLU, retrieval, RAG) means that future enhancements can be carried out without building the entire system again.

- Using open-source libraries (spaCy, Sentence Transformers, FastAPI) reduces dependency on a vendor for a long period.

Economic Sustainability

- The system reduces the level of manual work required of PGRKAM staff by automatically answering increasing basic queries.
- The low cost of operations makes it easier to deploy long-term in any government context.

8.5 Safety Aspects

Safety entails that the system will not output harmful, unsafe, or misleading information.

Safety Requirements

- **Unsafe Content Enforcement**
  - The system will make rule-based checks to screen out unsafe queries (i.e., violence, illegal, self-harm, etc.).
- **Content Safety (Punjabi and English)**
  - A bilingual safety filter will make it clear when learning ascribed to unsafe content even when the query is code-mixed.
- **Safety of Use Protections against Abuse of the System**
  - Rate limiting will help mitigate flooding or denial of service behavior.
- **Safety of Use in the Official Government Document Delivery System**
  - The system will not:
    - Promise the user work or guarantee work
    - Misrepresent eligibility criteria (i.e., veteran status)
    - Change the date when an official document is supposed to be due

Safety Risks

- User ambiguity in queries leads to risk for the user when responding unpredictably based on your queries.

- A user could solely rely upon the answers provided by the chatbot related to the explicitness of the official dispositions.

Safety Mitigation

- You should always provide a link to the original source of the official document in your final answer.
- When modifying your original query forms and responding to the user, mention if there is ambiguity in a given response to the user.
- You should periodically review moderated timestamps during log reviews for identifying patterns to analyze safety risk violations.

Chapter 9

Conclusion

The PGRKAM Smart Assistant (PSA) has proven to be a successful implementation and development of a Multi-Language Conversational Artificial Intelligence (AI) System, allowing Public Access to Government Employment Services in Punjab and providing users of a Web-Based Service a solution to the major challenges of Web-Based Services, such as overcoming Language Barriers, Simplifying Navigation through Complex Portals, eliminating the need for Real-Time User Assistance, etc. The PGRKAM Smart Assistant provides users with an easy, evidence-based, accurate, and timely response through three methods: Multilingual NLU, Hybrid Retrieval (BM25 + Vector Search), and Retrieval-Augmented Generation (RAG).

In evaluating the PGRKAM Smart Assistant, the results are characterized by high intent recognition accuracy, good entity extraction performance, excellent hybrid retrieval effectiveness, and a low hallucination level of generated responses by the PGRKAM Smart Assistant. The evaluation results from the PGRKAM Smart Assistant have also demonstrated very high performance levels in Punjabi, English, and Code-Mixed Queries. User testing results indicate that the PGRKAM Smart Assistant's conversational access reduces the search efforts needed to locate the efficiency in completing tasks, as compared to navigating a Web-Based Government Employment Services Portal manually.

The system's design is designed using the modular FastAPI architecture, which allows for scalability, maintainability, and secure deployment on the cloud, thus enabling the design of real-life public service systems. On the social level, it will improve digital inclusion, accessibility, and employment awareness for rural and semi-urban communities. The successful implementation of the PGRKAM Smart Assistant illustrates the practical application of AI-based conversational systems within e-Governance and public employment platforms.

References

[1] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.T., Rocktäschel, T. and Riedel, S., 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems, 33, pp.9459-9474.

[2] Gao, L., Ma, X., Lin, J. and Callan, J., 2023, July. Precise zero-shot dense retrieval without relevance labels. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 1762-1777).

[3] Sawarkar, K., Mangal, A. and Solanki, S.R., 2024. Blended RAG: Improving RAG Accuracy with Semantic Search and Hybrid Query-Based Retrievers. arXiv preprint arXiv:2404.07220.

[4] Kayamkulam Abdulkhader, N., 2025. Evaluating Retrieval Strategies for Domain-Specific RAG Systems through Automated Metrics and Human-Centered User Study.

[5] Rackauckas, Z., 2024. Rag-fusion: a new take on retrieval-augmented generation. arXiv preprint arXiv:2402.03367.

[6] Zaratiana, U., Tomeh, N., Holat, P. and Charnois, T., 2024, June. Gliner: Generalist model for named entity recognition using bidirectional transformer. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers) (pp. 5364-5376).

[7] Sohn, J., Jung, H., Cheng, A., Kang, J., Du, Y. and Mortensen, D.R., 2024. Zero-shot cross-lingual NER using phonemic representations for low-resource languages. arXiv preprint arXiv:2406.16030.

[8] Pattnayak, P., Patel, H. and Agarwal, A., 2025, May. Tokenization matters: Improving zero-shot ner for indic languages. In 2025 IEEE International Conference on Electro Information Technology (eIT) (pp. 456-462). IEEE.

[9] Murthy, R., Bhattacharjee, P., Sharnagat, R., Khatri, J., Kanojia, D. and Bhattacharyya, P., 2022. Hiner: A large hindi named entity recognition dataset. arXiv preprint arXiv:2204.13743.

[10] Gala, J., Chitale, P.A., AK, R., Gumma, V., Doddapaneni, S., Kumar, A., Nawale, J., Sujatha, A., Puduppully, R., Raghavan, V. and Kumar, P., 2023. Indictrans2: Towards high-quality and accessible machine translation models for all 22 scheduled indian languages. arXiv preprint arXiv:2305.16307.

[11] Javed, T., Bhogale, K., Raman, A., Kumar, P., Kunchukuttan, A. and Khapra, M.M., 2023, June. Indicsuperb: A speech processing universal performance benchmark for indian languages. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 37, No. 11, pp. 12942-12950).

[12] Singh, J. and Thakur, R., 2025. Quantum-RAG and PunGPT2: Advancing Low-Resource Language Generation and Retrieval for the Punjabi Language. arXiv preprint arXiv:2508.01918.

[13] Tripathi, K., Gothi, R. and Wasnik, P., 2025, April. Enhancing Whisper's accuracy and speed for Indian languages through prompt-tuning and tokenization. In ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1-5). IEEE.

[14] Varghese Thomas, FastAPI for Scalable Microservices: Architecture, Optimisation and Best Practices, Webandcrafts, 2024

[15] Papadopoulos, T., Alexopoulos, C. and Charalabidis, Y., 2025. Evaluating chatbot architectures for public service delivery: balancing functionality, safety, ethics, and adaptability. Frontiers in Political Science, 7, p.1601440.

[16] Bucaioni, A., Weyssow, M., He, J., Lyu, Y. and Lo, D., 2025. Artificial Intelligence for Software Architecture: Literature Review and the Road Ahead. arXiv preprint arXiv:2504.04334.

[17] Devanshu Agarwal, Dr. Vishal Shrivastava, Dr. Akhil Pandey, Dr. Karuna Sharma, Dr. Ashok Kumar

Kajla, Building Scalable Web Applications with Python , IJRPR Volume 6, ISSUE 5, May 2025, Page NO:

6591-6597.

[18] Es, S., James, J., Anke, L.E. and Schockaert, S., 2024, March. Ragas: Automated evaluation of retrieval augmented generation. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations (pp. 150-158).

[19] Niu, C., Wu, Y., Zhu, J., Xu, S., Shum, K., Zhong, R., Song, J. and Zhang, T., 2024, August. Ragtruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 10862-10878).

[20] Ahmad, S., Nezami, Z., Hafeez, M. and Zaidi, S.A.R., 2025. Benchmarking Vector, Graph and Hybrid Retrieval Augmented Generation (RAG) Pipelines for Open Radio Access Networks (ORAN). arXiv preprint arXiv:2507.03608.

[21] Krishna, S., Krishna, K., Mohananey, A., Schwarcz, S., Stambler, A., Upadhyay, S. and Faruqui, M., 2025, April. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers) (pp. 4745-4759).

[22] Barnett, S., Kurniawan, S., Thudumu, S., Brannelly, Z. and Abdelrazek, M., 2024, April. Seven failure points when engineering a retrieval augmented generation system. In Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI (pp. 194-199).

Base Paper

As a foundational reference for establishing the foundation for its PGRKAM Smart Assistant, we chose the paper by Patrick Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" from Facebook AI Research in 2020 that outlines a framework to build an accurate and credible response to the user's question through the combination of document retrieval and generation through the RAG model (retrieval as an augmentation for generation).

Paper Link: https://doi.org/10.48550/arXiv.2005.11401

Appendix A

Research Paper Submission

Paper Information

Journal	SRP Journals (SRP)
Paper ID	2146017
Paper Title	A Unified Multilingual Smart Assistant for PGRKAM: Combining Speech Input, Language Detection, and AI-Powered Job Query Understanding
Keywords	Multilingual Chatbots, Employment Portals, Retrieval-Augmented Generation, Code-Switching, Dialogue State Tracking, Compliance, Public Sector AI
Abstract	There are many barriers for government job seekers in linguistically diverse regions (such as Punjab) including language disparity and complexity of information, which often leads to lack of awareness in job seekers. Traditional digital solutions have been unable to address such problems as they often fail to resolve the ambiguity, provide poor multilingual support and provide inaccurate context-based answers for noisy user inputs. This paper addresses these challenging problems by proposing a novel, unified multilingual smart assistant called PGRKAM Smart Assistant to make government job schemes and application processes accessible in an efficient manner. Our approach utilizes a Multilingual Hybrid Retrieval Augmented Generation Chatbot architecture with state-of-the-art solutions for speech-assisted input processing, language detection and ad vanced job query understanding using Artificial Intelligence. The proposed architecture leverages a multi-phase retrieval pipeline, dynamic dialogue state tracking and specific training with lo calized multilingual datasets to address information overload and language-specific understanding issues. Preliminary exper iments demonstrate that the proposed architecture achieves 90% intent recognition accuracy with GLiNER and exhibits better understanding of complex job queries in several languages. The unified architecture facilitates equitable and efficient information access for all government job seekers by addressing the critical challenges associated with linguistic diversity.
Fields	Computer Science & Communications

Figure A - Research Paper Submission Image

Appendix B

Datasets

PGRKAM Job Notifications Dataset (Official Portal Extraction)

- Samples: ~500 job postings
- Time period: January 2022 – October 2025
- Variables: Job title, department, employer name, job type (government/private), qualification, age limit, gender eligibility, salary range, location, last date, application link
- Target variables: Job category, eligibility status, matching relevance score
- Format: JSON
- Source: Official PGRKAM public job listings (web-extracted)

PGRKAM Government Schemes & Training Programs Dataset

- Samples: ~200 scheme and training records
- Time period: Ongoing (latest updates till October 2025)
- Variables: Scheme name, department, beneficiary category, eligibility criteria, benefits, application procedure, supporting documents, deadlines
- Target variables: Scheme relevance, eligibility match
- Format: JSON
- Source: Official PGRKAM and Punjab Government scheme portals

Employment FAQs & Helpdesk Interaction Dataset

- Samples: ~500 question–answer pairs
- Time period: 2021 – 2025
- Variables: User question, system answer, category, related job/scheme reference
- Target variables: Intent label, FAQ category
- Format: CSV
- Source: Public employment help portals and PGRKAM support pages

Appendix C

Github Repository

Github Repository Link: https://github.com/ashwin2947/PGRKAM-Smart-Assistant

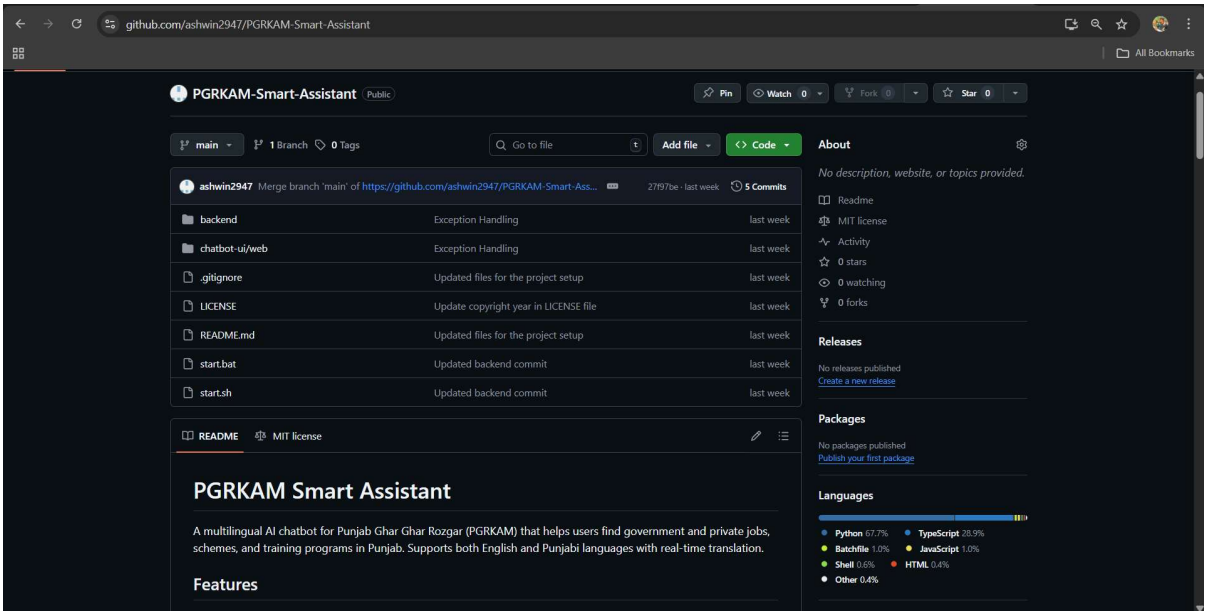

Figure B - Project Github Repository

Appendix C

Chatbot Interface Images

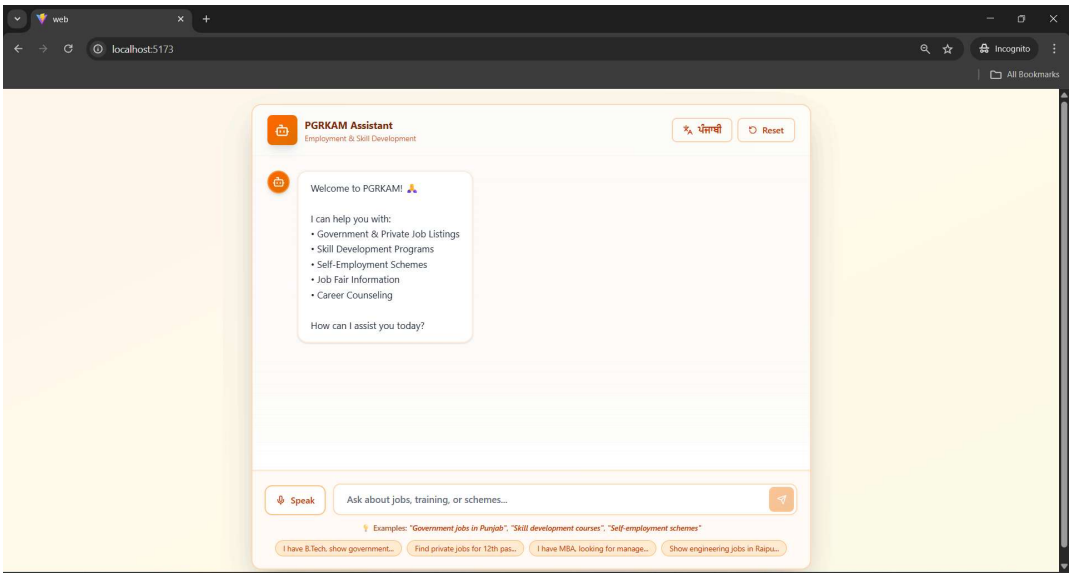

Figure C - Chatbot Web Interface

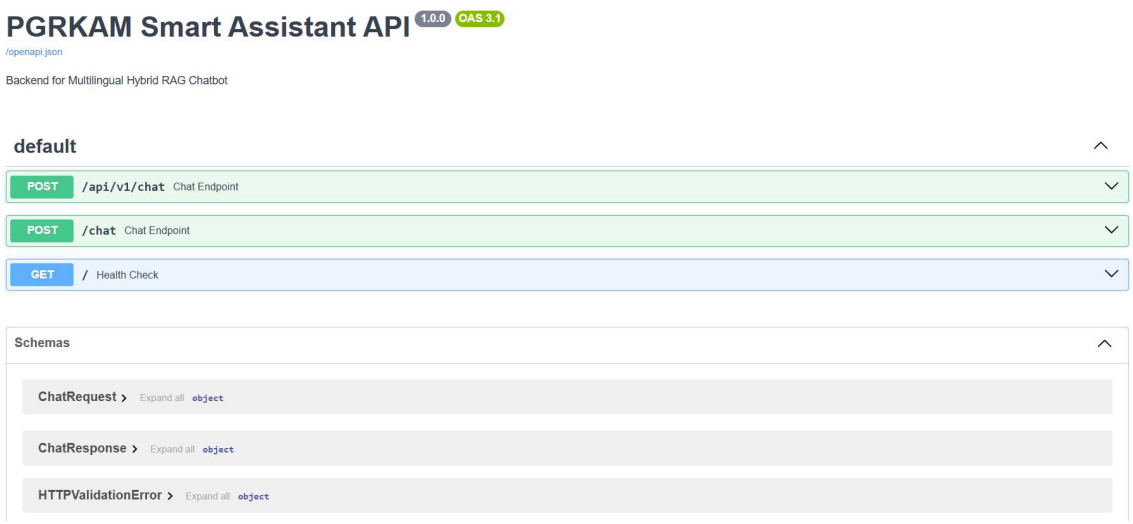

Figure D - Chatbot API Interface

Appendix D

Turnitin Report

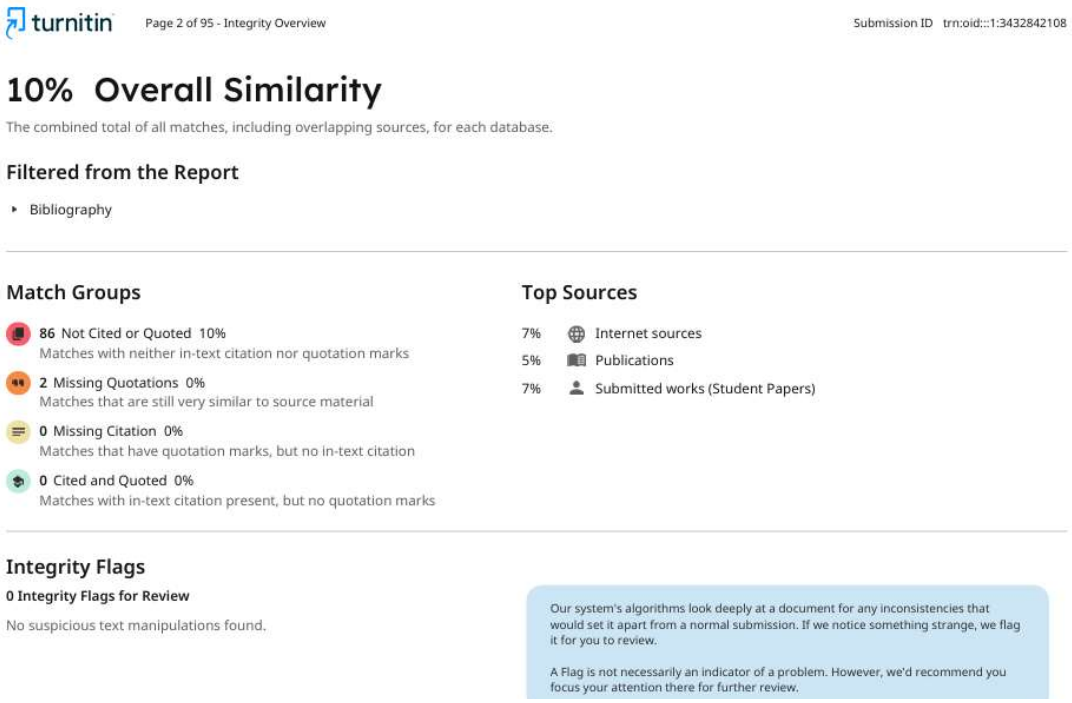

Figure E - Report Similarity Percentage (%)

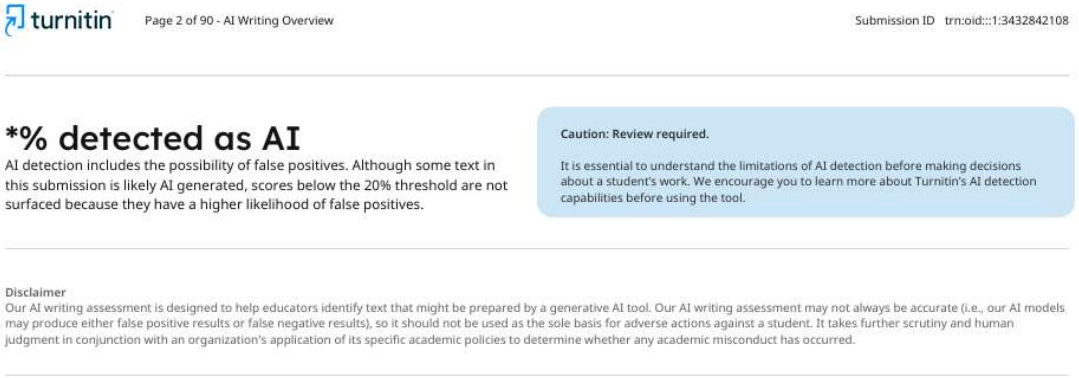

Figure F - Report AI Percentage (%)