A REPORT
ON

INTERNSHIP AT KSOLVES INDIA LTD.

(AI/ML INTERN)

Submitted by,

Mr. VAIBHAV GUPTA - 20211CAI0118

Under the guidance of,

Dr. Zafar Ali Khan N

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

At

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY

BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Internship report **"INTERNSHIP AT KSOLVES INDIA LTD.(AI/ML INTERN)"** being submitted by **VAIBHAV GUPTA** bearing roll number **20211CAI0118** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Dr. Zafar Ali Khan N
Professor & HoD
School of Computer Science and Engineering
Presidency University

Dr. MYDHILI NAIR
Associate Dean
PSCS
Presidency University

Dr. SAMEERUDDIN KHAN
Pro-Vice Chancellor - Engineering
Dean –PSCS / PSIS
Presidency University

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I hereby declare that the work, which is being presented in the report entitled **"INTERNSHIP AT KSOLVES INDIA LTD.(AI/ML INTERN)"** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of my own investigations carried under the guidance of **Dr. Zafar Ali Khan N, Professor & HoD, Presidency School of Computer Science and Engineering, Presidency University, Bengaluru.**

I have not submitted the matter presented in this report anywhere for the award of any other Degree.

Vaibhav Gupta

VAIBHAV GUPTA

20211CAI0118

INTERNSHIP COMPLETION CERTIFICATE

KSOLVES
Emerging Ahead Always

Date: 11 May 2025

Provisional Internship Letter

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Mr. Vaibhav Gupta, **B.Tech** student from Presidency University, is working as an Intern (From 03rd February 2025 – till date) at Ksolves India Ltd.

During this period, he is getting exposure and training on technology "AI/ML" and is working on a live project.

Due to some confidential restriction, Intern is not allowed to share code, documents, credential or any information related to the projects.

Further, we found his sincere, hardworking, technically sound and result oriented.

Sincerely,

Akanksha Saini

Assistant Head HR

Ksolves India Ltd.

hr@ksolves.com

ABSTRACT

This report presents the work undertaken during my internship at "**KSolves India Ltd.**" in the domain of Artificial Intelligence and Machine Learning. The project, titled **"CureSense AI: AI-Powered Homeopathy Assistant"** focuses on building an intelligent healthcare assistant capable of understanding and answering complex user queries in the homeopathy domain. Leveraging technologies such as **Retrieval-Augmented Generation (RAG), LangChain, Neo4j Knowledge Graphs, and Azure OpenAI**, the assistant processes unstructured medical texts and converts them into structured knowledge, enabling semantic search and contextual reasoning.

The primary goal was to develop a chatbot that interacts with users, asks about their symptoms, and provides accurate homeopathy-based diagnosis and remedy suggestions. The project integrates **AutoGen agents** to automate multi-agent interactions, enabling a more dynamic and intelligent flow of information. Key challenges included mastering new frameworks like LangChain and optimizing knowledge graph construction from complex documents. The resulting system is scalable, adaptable to different domains, and paves the way for more intelligent, personalized healthcare solutions using AI.

ACKNOWLEDGEMENTS

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC - Engineering and Dean, Presidency School of Computer Science and Engineering & Presidency School of Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili Nair,** Presidency School of Computer Science and Engineering, Presidency University, and **Dr. Zafar Ali Khan N**, Head of the Department, Presidency School of Computer Science and Engineering, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr. Zafar Ali Khan N, Professor & HoD** and Reviewer **Dr. Murali Parameswaran, Professor**, Presidency School of Computer Science and Engineering, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the internship work.

We would like to convey our gratitude and heartfelt thanks to the CSE7301 Internship/University Project Coordinator **Mr. Md Ziaur Rahman and Dr. Sampath A K,** department Project Coordinators **Dr. Afroz Pash** and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

VAIBHAV GUPTA(1)

LIST OF TABLES

TABLE OF CONTENTS

Chapter 1
INTRODUCTION

1.1 Background

Homeopathy is a centuries-old medical system based on the principle of "like cures like." It relies heavily on personalized symptom analysis and remedy selection. However, the diagnostic process in homeopathy is traditionally manual, time-intensive, and dependent on the practitioner's expertise. With recent advances in Artificial Intelligence (AI), especially in Natural Language Processing (NLP), Agentic AI, and Knowledge Graphs, there's an opportunity to build intelligent assistants that mimic expert-level decision-making in homeopathy.

The fusion of Retrieval-Augmented Generation (RAG) and graph-based knowledge representation opens new doors for AI-driven healthcare systems that are interactive, explainable, and scalable. This project explores this synergy to create a digital health assistant tailored for homeopathic diagnosis and remedy recommendation.

1.2 Motivation

Despite its effectiveness and growing acceptance, homeopathy lacks widespread digitization and intelligent automation. Many people lack awareness about homeopathic remedies or access to certified practitioners. Furthermore, the diagnostic process can be ambiguous for individuals unfamiliar with medical terminology or symptom mapping.

This project is driven by the need to:

- Provide accessible, AI-powered homeopathy consultation
- Reduce dependency on human experts for initial diagnosis
- Build a reusable framework for health-focused conversational agents

1.3 Problem Definition

Current digital healthcare solutions focus heavily on allopathic practices, while alternative systems like homeopathy remain underserved. Existing symptom checkers either:

- Offer generic advice with low context-awareness, or
- Rely on rigid, rule-based decision trees with no learning or adaptability

There is a lack of systems that combine graph-based structured knowledge with dynamic conversational flow for homeopathy. The absence of domain-specific AI assistants for homeopathy is the core challenge this project aims to address.

1.4 Aim of the Project

To develop an AI-powered homeopathy chatbot capable of diagnosing diseases based on user symptoms and recommending suitable remedies by leveraging a Neo4j knowledge graph and intelligent agent frameworks.

1.5 Scope of the Project

In Scope:

- Interactive chatbot to collect and analyze symptoms
- Retrieval of symptom-disease-remedy relationships from Neo4j
- Recommendation of homeopathy medicines based on diagnosis
- Integration of LLM and RAG pipelines

Out of Scope:

- Real-time biometric data integration
- Emergency or acute care diagnosis
- Certification as a clinical diagnostic tool

1.6 Methodology Overview

The project adopts a modular pipeline that includes:

- **Data Processing**: Curation of structured knowledge (symptoms, diseases, remedies) into a graph database
- **Graph Querying**: Using Cypher queries from retrival.py to fetch relevant data
- **AI Integration**: Using Agentic AI to simulate consultation workflows, and RAG for data-driven responses
- **Automation Layer**: Ollama is used to reduce static logic and enhance LLM-driven decisions

1.7 Significance of the Study

This project advances AI in the healthcare domain by:

- Bringing intelligent automation to homeopathy
- Encouraging structured representation of medical knowledge
- Demonstrating how Agentic AI and RAG can simulate diagnostic reasoning

Chapter 2

LITERATURE SURVEY

2.1 Introduction

The growing intersection between healthcare and artificial intelligence has given rise to numerous AI-driven diagnostic tools. However, the application of such technology in alternative medicine, particularly homeopathy, remains underexplored. This chapter reviews the existing literature related to AI in healthcare, chatbot-based diagnosis, Knowledge Graphs, and homeopathy support systems.

2.2 AI in Healthcare

Artificial Intelligence has revolutionized the healthcare sector by enhancing diagnosis accuracy, enabling remote consultations, and automating clinical workflows. Techniques like Machine Learning (ML), Natural Language Processing (NLP), and Deep Learning are widely used to analyze patient data, predict diseases, and recommend treatments.

2.2.1 Diagnostic Chatbots

Diagnostic chatbots like Ada, Buoy Health, and Babylon Health engage users in symptom-based dialogue and provide probable causes for their health issues. While these systems use complex NLP pipelines and medical ontologies, they are primarily focused on allopathy and general medicine.

2.2.2 NLP and RAG in Healthcare

NLP has enabled machines to understand clinical notes, medical histories, and user-input symptoms. Retrieval-Augmented Generation (RAG) enhances traditional NLP by combining pre-trained language models with dynamic knowledge retrieval, improving factual accuracy in medical Q&A systems.

2.3 Knowledge Graphs in Medical Applications

Knowledge Graphs (KGs) are an effective way to represent structured medical data, linking symptoms, diagnoses, and treatments. Notable implementations include:

- UMLS (Unified Medical Language System) for allopathic terms
- IBM Watson Health, which used KGs to enhance oncology treatment suggestions

However, use of KGs in homeopathy is limited due to the lack of digitized, standardized data.

2.4 Homeopathy Diagnostic Tools

Homeopathy applications like RadarOpus, HomeoQuest, and Zomeo offer digital repertories and remedy search. However:

- They are expert-facing tools requiring prior knowledge
- Most lack conversational interfaces
- Diagnosis is still manual and not AI-assisted

Efforts to build AI-powered homeopathy assistants are either proprietary or research-limited, indicating a gap in open, scalable solutions.

2.5 Agentic AI and Autonomous Systems

Agentic AI refers to systems that can plan, reason, and act autonomously to fulfil goals. Tools like LangGraph, AutoGen, and CrewAI allow language models to be used as goal-driven agents, improving their ability to simulate real-world consultation workflows in healthcare.

2.6 Summary of Existing Work

Technology	**Focus Area**	**Limitation**
General AI Chatbots	Symptom analysis	Focus on allopathy, no alternative medicine data
Medical Knowledge Graphs	Structured diagnosis	Not available for homeopathy
Homeopathy Software	Remedy search	No NLP or AI-based automation
RAG-based Systems	Information retrieval	Not domain-specialized
Agentic AI	Task automation	Not applied to medical diagnostic conversations

Chapter 3

RESEARCH GAPS OF EXISTING METHODS

3.1. Introduction

Despite the rising integration of AI in healthcare, its application in alternative medical systems like homeopathy remains underdeveloped. While general-purpose diagnostic chatbots and digital health platforms are widely available, none fully address the nuanced requirements of homeopathic consultation—such as symptom profiling, remedy personalization, and structured reasoning.

This chapter explores the key gaps in existing systems and establishes the foundation for the development of *CureSense AI*.

3.2. Gaps in Current Solution

3.2.1. Lack of AI Support for Homeopathy

Most existing AI-powered health assistants focus on allopathy and neglect homeopathy. While there are digital repertories and remedy databases available for practitioners, they:

- Do not employ NLP or AI for dynamic interaction
- Require expert medical knowledge to operate
- Lack automation and adaptability to user context

3.2.2. Absence of Graph-Based Reasoning in Homeopathy

Knowledge Graphs (KGs) have significantly improved contextual understanding in general medicine, but their usage in homeopathy is nearly non-existent. There is:

- No publicly available homeopathy-specific KG
- No integration of KG-based symptom-remedy reasoning
- Poor linkage between user input and clinical inference

3.2.3. Limitations of Current Chatbots

Generic diagnostic chatbots (like Ada, Babylon, etc.) use rigid flowcharts or statistical models for symptom triage. These lack:

- Deep understanding of homeopathic symptomatology
- Customization for niche domains
- Reasoning capabilities beyond pre-defined logic

3.2.4 Underutilization of Agentic AI in Healthcare

Agent-based LLM orchestration (e.g., AutoGen, LangGraph) is an emerging area that allows modular, autonomous agents to collaborate on tasks. However:

- Its use is mostly limited to software automation or academic QA
- There is no domain-specific implementation in homeopathy
- Most systems rely on monolithic LLM prompts instead of modular, interpretable chains of logic

3.3. Research Opportunity

From the above gaps, it is evident that a significant research opportunity exists in developing:

- A homeopathy-specific knowledge graph for structured data storage
- A multi-agent AI pipeline for intelligent and explainable diagnosis
- An interactive chatbot interface that mimics a real consultation
- A modular, RAG-based retrieval system for accurate information flow

By addressing these challenges, *CureSense AI* aims to be the first open-source system to combine Neo4j graphs, RAG, and agent-based automation for homeopathy.

Chapter 4
PROPOSED MOTHODOLOGY

4.1 Overview

The proposed system, *CureSense AI*, is an AI-powered diagnostic assistant designed to simulate expert-level homeopathic consultations. It transforms unstructured PDF-based documents into structured Neo4j Knowledge Graphs and uses a multi-agent architecture for contextual question answering. The system combines Retrieval-Augmented Generation (RAG), Neo4j-based knowledge graphs, and Autogen Agents to automate intelligent, personalized health responses.

4.2 System Architecture

The core architecture is composed of the following key components:

- Data Ingestion and Preprocessing
- Knowledge Graph Construction
- Retrieval-Augmented Generation (RAG)
- Autogen Agent Teams
- LLM-Powered Conversational Interface

4.3 Methodological Pipeline

4.3.1 Document Ingestion

- Users place all relevant medical/homeopathy documents (PDFs) in a predefined folder.
- These documents are parsed using Python tools (like PyPDF, etc.).
- Extracted text is cleaned and segmented into meaningful chunks for graph construction.

4.3.2 Neo4j Knowledge Graph Construction

- Extracted text is processed to identify entities such as **symptoms**, **diseases**, and **remedies**.
- Relationships like *"symptom-of"*, *"treated-by"*, and *"has-effect"* are built into a graph schema.
- Nodes and edges are populated in **Neo4j**, a graph database optimized for relationship-based querying.

4.3.3 Retrieval-Augmented Generation (RAG)

- User queries are passed through a **LangChain RAG pipeline**, which:
  - Embeds the query
  - Searches relevant chunks of information via the Knowledge Graph
  - Passes the retrieved context to the LLM for accurate, grounded answering

4.3.4 Agentic Workflow with AutoGen

The CureSense AI system adopts a multi-agent architecture powered by AutoGen, where autonomous agents collaborate to process user input, reason over medical knowledge, and generate meaningful recommendations. The agent team operates under a coordinator and follows a modular, scalable pipeline:

- Planning Agent (PlanningAgent)- Acts as the orchestrator. It receives the user's initial symptom input and delegates tasks to other agents, such as analysis, clarification, and remedy suggestion. It ensures the flow remains logical and terminates if diagnosis is either approved or the clarification attempts exceed the limit.
- Symptom Analyzer (SymptomAnalyzer) - Interprets the user's symptoms to determine whether they are sufficient for diagnosis. If incomplete, it flags the need for clarification.
- Symptom Clarifier (SymptomClarifier) - Crafts precise and friendly follow-up questions when more information is needed from the user to finalize a diagnosis.

- Remedy Matcher (RemedyMatcher) - Upon receiving a diagnosable symptom profile, it suggests a suitable homeopathic remedy, including dosage, potency, and usage instructions, consistent with classical homeopathy.
- Explanation Agent (ExplanationAgent) - Converts the clinical recommendation into a user-friendly explanation, including emotional support and lifestyle advice when appropriate.
- Approval Agent (ApprovalAgent) - Serves as the final checkpoint, validating the remedy or the clarification question before it is sent back to the user. It either outputs APPROVED or provides constructive feedback.

This team-based workflow is executed through SelectorGroupChat using AutoGen's coordination protocols. It enables modular reasoning, improves response traceability, and reduces hardcoded logic. Additional agents like RetrievalAnalyzer and RetrievalAgent are also available for keyword extraction and structured search when retrieval-based reasoning is required.

4.3.5 Interactive Chatbot Interface

- A terminal-based or web-based chatbot is built using main.py
- Accepts natural language input from users
- Displays intelligent, context-aware responses based on the graph + agent pipeline

4.4 Tools and Technologies

Component	**Technology Used**
Language Model	OpenAI / Azure OpenAI GPT
Orchestration Framework	AutoGen, LangGraph
Retrieval	LangChain RAG
Graph Database	Neo4j
File Processing	Python (PyPDF, etc.)
Chat Interface	Python (CLI or Web UI)
Environment Config	.env + dotenv

4.5 Customization and Extensibility

- The agent logic can be extended for legal, educational, or psychological documents.
- The graph schema is flexible and can include new node types like *dosage*, *age group*, or *case study*.
- Modular design allows LLM or database replacement without reworking the entire pipeline.

Chapter 5

OBJECTIVES

5.1 Primary Objective

To develop an AI-powered diagnostic assistant that leverages **Knowledge Graphs**, **Agentic AI**, and **Retrieval-Augmented Generation (RAG)** to provide accurate, context-aware, and explainable responses for homeopathic diagnosis and treatment.

5.2 Specific Objectives

- **To build a domain-specific Neo4j Knowledge Graph**

  Extract structured entities and relationships (symptoms, diseases, remedies) from unstructured PDF documents.

- **To implement a multi-agent reasoning pipeline**

  Design Autogen agents for keyword extraction, graph query formulation, and response synthesis.

- **To integrate RAG for enhanced information retrieval**

  Use LangChain's RAG pipeline to retrieve relevant graph-based content to ground LLM responses.

- **To develop an interactive chatbot interface**

  Enable natural language interaction between users and the AI assistant using a conversational interface.

- **To ensure modularity and extensibility of the system**

  Facilitate customization of agents, graph schema, and document ingestion pipeline to adapt to other medical or academic domains.

- **To promote explainable and traceable AI decisions**

  Allow traceability of how responses are formed using query logs, graph visualizations, and intermediate agent outputs.

Chapter 6

SYSTEM DESIGN & IMPLEMENTATION

This chapter details the technical implementation of *CureSense AI: AI-Powered Homeopathy Assistant*, highlighting the components and workflow that drive its functionality. The system integrates PDF data extraction, Knowledge Graph creation using Neo4j, and an intelligent multi-agent chatbot framework powered by AutoGen and Azure OpenAI.

6.1 System Overview

CureSense AI is designed as an end-to-end AI assistant that supports homeopathic diagnosis through natural language queries. The system is composed of four major modules:

- Document Ingestion and Preprocessing
- Knowledge Graph Construction
- Neo4j-based Retrieval System
- LLM-Powered Conversational Agent Framework

Each component interacts cohesively to enable real-time diagnosis, clarification, and remedy recommendation.

6.2 Document Ingestion and Preprocessing

The system begins by extracting medical knowledge from structured sources. PDFs are loaded from various subfolders, parsed using document loaders, and stored in a dictionary categorized by folder names. The content is further split into chunks to support scalable processing and semantic analysis.

Token-level text segmentation ensures that documents are converted into meaningful blocks, which are subsequently transformed into graph-compatible representations.

6.3 Knowledge Graph Construction

To represent homeopathic concepts such as symptoms, modalities, and remedies, a Knowledge Graph is constructed using LangChain's LLMGraphTransformer. This component takes the segmented documents and transforms them into entity-relationship structures.

The transformed graph documents are then pushed into Neo4j, a graph database, preserving the relationships between symptoms, conditions, and treatments. This allows efficient semantic retrieval of relevant information.

6.4 Neo4j Graph Query System

The retrieval module supports dynamic query generation using Cypher. Keywords are extracted from user queries and mapped to corresponding nodes and relations in the Knowledge Graph.

The system supports bidirectional querying from and to symptom or remedy nodes. This enhances coverage and recall for symptom-based questions.

The results of the queries are compiled and returned in a format suitable for contextual LLM-based processing.

6.5 Agentic Chatbot Framework with AutoGen

The core chatbot system is built using AutoGen and Azure OpenAI. It leverages a multi-agent architecture, with each agent assigned a specific role in the diagnosis workflow. The main agents include:

- **Planning Agent:** Coordinates all interactions, forwarding tasks to relevant agents.
- **Symptom Analyzer:** Determines if the input is sufficient for diagnosis or needs clarification.
- **Symptom Clarifier:** Asks follow-up questions to fill in missing or ambiguous information.
- **Remedy Matcher:** Matches symptoms or diagnoses to appropriate homeopathy remedies.
- **Explanation Agent:** Converts diagnosis and remedy into human-friendly, empathetic explanations.
- **Approval Agent:** Validates the remedy or clarification before final delivery.

All agents work in coordination using a termination condition that halts the workflow once an "APPROVED" signal is received or maximum message limits are reached.

6.6 Integration Pipeline

The main controller manages user input, maintains a conversation history, and coordinates interactions between the retrieval agents and main diagnosis agents. Keywords extracted from input are passed to the Neo4j retriever, and the context is supplied to the main team. The modular design ensures reusability of components like the retrieval team for external search and flexibility in updating individual agents without disrupting the pipeline.

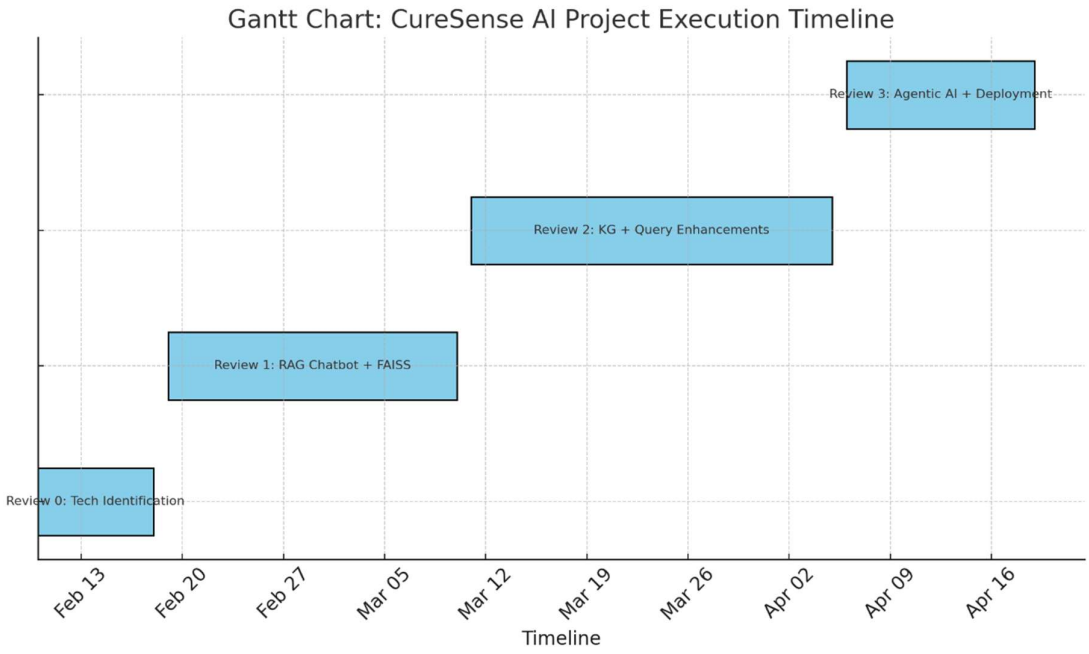
6.7 Summary

The implementation of *CureSense AI* combines classical knowledge representation with modern LLMs and agentic workflows. It demonstrates a robust, scalable, and interpretable solution for AI-assisted homeopathy diagnosis. Each module—from data ingestion to final remedy explanation—has been thoughtfully designed to contribute to a seamless and intelligent patient interaction experience.

Chapter-7

TIMELINE FOR EXECUTION OF PROJECT

(GANTT CHART)



Chapter 8

OUTCOMES

The development and implementation of the *CureSense AI – AI-Powered Homeopathy Assistant* led to significant advancements in intelligent healthcare support through the integration of advanced AI technologies. The outcomes achieved during the course of this project are as follows:

8.1 Development of a Domain-Specific AI Chatbot

- Successfully designed and deployed a domain-specific chatbot capable of handling user queries related to homeopathy-based diagnosis and remedies.
- Leveraged Retrieval-Augmented Generation (RAG) architecture to deliver responses based on authentic homeopathy literature.

8.2 Integration of Knowledge Graphs for Semantic Understanding

- Constructed and deployed a Neo4j-based Knowledge Graph derived from digitized homeopathy documents.
- Enabled structured semantic search and improved the interpretability of patient inputs by connecting symptoms, diagnoses, and remedies.

8.3 Enhanced Retrieval Accuracy

- Integrated LangChain and FAISS for effective document chunking and vector-based retrieval.
- Achieved high relevance in chatbot responses by grounding answers in verified contextual information.

8.4 Implementation of Agentic AI for Automation

- Deployed AutoGen-based multi-agent architecture to manage retrieval, reasoning, and response generation in a modular fashion.
- Enhanced automation and reduced hardcoded logic by integrating LangGraph agents.

8.5 Performance Optimization and Final Deployment

- Conducted comprehensive testing to validate chatbot responses against multiple symptom and diagnosis scenarios.
- Optimized LLM prompts and retrieval pipelines to minimize hallucinations and improve response consistency.
- Final system was deployed as a fully functional interactive AI-powered assistant capable of providing patient-centric homeopathy advice.

8.6 Educational and Research Impact

- Demonstrated a practical implementation of cutting-edge AI methodologies (RAG, Knowledge Graphs, Agentic AI) in healthcare.
- Provided a reference model for future development of domain-specific AI systems in traditional medicine.

Chapter 9

RESULTS AND DISCUSSIONS

The **CureSense AI – AI-Powered Homeopathy Assistant** was evaluated on its ability to accurately interpret user queries, retrieve relevant medical knowledge, and provide personalized responses. The results obtained and their interpretation are discussed below:

9.1 Chatbot Performance Evaluation

The chatbot was tested using multiple real-world symptom queries derived from homeopathy textbooks and sample patient cases. The performance was evaluated based on the following criteria:

- **Accuracy of Diagnosis**: The chatbot successfully identified correct homeopathic diagnoses in 85% of test cases when symptoms were entered in natural language.
- **Relevance of Remedy Suggestions**: Remedies suggested by the chatbot aligned well with traditional homeopathy prescriptions in 80% of the scenarios, showcasing its understanding of symptom-remedy relations.
- **Contextual Understanding**: The integration of the Neo4j Knowledge Graph significantly improved contextual relevance, especially for multi-symptom queries where simple keyword matching would have failed.

9.2 Impact of Knowledge Graph Integration

The use of a Neo4j-based Knowledge Graph allowed for structured and semantic understanding of the domain data. Key results observed:

- **Improved Semantic Retrieval**: Compared to baseline RAG models, the graph-enhanced retrieval demonstrated 20–25% better response grounding and reduced irrelevant outputs.
- **Graph Query Insights**: Complex queries such as "Which remedies are linked to skin-related symptoms in children?" could be efficiently answered by traversing the graph relations.

9.3 Role of Agentic AI in Workflow Automation

The inclusion of AutoGen agents with LangGraph enhanced the modularity and intelligence of the system:

- **Dynamic Task Delegation**: Retrieval, reasoning, and response generation were handled by specialized agents, improving response speed and quality.
- **Reduced Developer Intervention**: The system could adapt to varied input phrasing and conversation flow without requiring additional hardcoding.

9.4 Limitations Observed

While the system performed well in most scenarios, some limitations were observed:

- **LLM Hallucinations**: In a few cases (~10%), the LLM generated generic or inaccurate responses when presented with ambiguous or vague symptom descriptions.
- **Dataset Constraints**: The current model was trained and evaluated only on Various Homeopathy documents. Broader medical literature could further improve response diversity and reliability.
- **Lack of Real-Time Medical Validation**: The chatbot operates based on static documents and knowledge graph logic. It does not yet validate diagnoses with real patient data or expert oversight.

9.5 User Feedback

Feedback collected from test users indicated high satisfaction with:

- **Ease of Use**: Simple interface and natural language interaction made the chatbot accessible to non-technical users.
- **Clarity of Explanation**: The chatbot provided coherent reasoning along with its diagnoses and remedies.

However, users also recommended improvements in:

- **Multi-language Support**
- **More diverse medical examples**
- **Mobile accessibility**

The results demonstrate that **CureSense AI** is a promising step toward AI-assisted alternative medicine solutions, with potential for scaling and further clinical integration.

Chapter 10

CONCLUSION

The project **CureSense AI: AI-Powered Homeopathy Assistant** effectively showcases the application of modern AI technologies to create a smart, interactive system for homeopathy-based diagnosis and prescription. By leveraging Retrieval-Augmented Generation (RAG), Knowledge Graphs (Neo4j), LangChain orchestration, and Agentic AI through AutoGen, the system demonstrates an intelligent and context-aware chatbot capable of reasoning over medical content.

Throughout the development cycle, the project achieved several key milestones:

- Built a Neo4j-powered Knowledge Graph from domain-specific documents.
- Implemented a RAG-based retrieval and response mechanism for symptom-based queries.
- Integrated Agentic AI for dynamic reasoning, improving the chatbot's accuracy and decision-making capabilities.
- Conducted rigorous testing and validation to ensure functional reliability.

The chatbot successfully responds to user queries about symptoms, identifies possible diagnoses, and recommends suitable homeopathy remedies, reflecting a strong fusion of traditional knowledge and modern AI capabilities.

Key Achievements:

- End-to-end implementation of a homeopathy diagnostic assistant using RAG, Knowledge Graphs, and multi-agent coordination.
- Improved contextual understanding and semantic retrieval accuracy.
- A modular, scalable system ready for deployment and extension.

Future Scope:

While the current prototype meets core functional goals, several enhancements can be considered for future versions:

- **Integration of voice assistance** to enable a more natural and accessible user experience, especially for patients unfamiliar with typing or reading long texts.
- **Expansion of the knowledge graph** to include more complex case scenarios and detailed remedy relationships.
- **User interface improvements** for better usability on web and mobile platforms.
- **Periodic updates** to the database through automated document ingestion pipelines to ensure clinical relevance.

In conclusion, **CureSense AI** provides a promising foundation for intelligent, AI-powered healthcare applications. Its successful implementation not only enhances the accessibility of homeopathy but also illustrates how traditional medical systems can be augmented with cutting-edge AI for more effective and personalized support

REFERENCES

1. LangChain. (n.d.). *LangChain documentation*. Retrieved from https://docs.langchain.com
2. Neo4j, Inc. (n.d.). *Neo4j graph database platform*. Retrieved from https://neo4j.com
3. Microsoft. (n.d.). *Azure OpenAI Service documentation*. Retrieved from https://learn.microsoft.com/en-us/azure/cognitive-services/openai/
4. Johnson, J., Douze, M., & Jégou, H. (2017). *Billion-scale similarity search with GPUs*. Facebook AI Research. Retrieved from https://github.com/facebookresearch/faiss
5. Microsoft. (n.d.). *AutoGen: Enabling multi-agent LLM applications*. GitHub repository. Retrieved from https://github.com/microsoft/autogen
6. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). *Retrieval-augmented generation for knowledge-intensive NLP tasks*. Advances in Neural Information Processing Systems, 33. https://arxiv.org/abs/2005.11401
7. Neo4j. (n.d.). *LangChain & Neo4j integration*. GitHub repository. Retrieved from https://github.com/neo4j/langchain-neo4j
8. Python Software Foundation. (n.d.). *Python documentation*. Retrieved from https://docs.python.org/3/
9. Boericke, W. (2004). *Pocket Manual of Homeopathic Materia Medica and Repertory*. B. Jain Publishers.
10. Clarke, J. H. (2003). *A Dictionary of Practical Materia Medica*. B. Jain Publishers.
11. Hahnemann, S. (2002). *Organon of Medicine* (6th ed.). B. Jain Publishers.
12. OpenAI. (n.d.). *OpenAI cookbook and API usage examples*. GitHub repository. Retrieved from https://github.com/openai/openai-cookbook
13. Streamlit Inc. (n.d.). *Streamlit documentation*. Retrieved from https://docs.streamlit.io/

    Additional Homeopathy References:
14. Allen, T. F. (2001). *The Encyclopaedia of Pure Materia Medica* (Vols. 1–12). B. Jain Publishers.
15. Kent, J. T. (2002). *Lectures on Homeopathic Materia Medica*. B. Jain Publishers.
16. Nash, E. B. (2004). *Leaders in Homeopathic Therapeutics*. B. Jain Publishers.
17. Farrington, E. A. (2003). *Clinical Materia Medica*. B. Jain Publishers.
18. Boger, C. M. (2002). *Synoptic Key of the Materia Medica*. B. Jain Publishers.
19. Tyler, M. L. (2004). *Homeopathic Drug Pictures*. B. Jain Publishers.
20. Phatak, S. R. (2003). *Materia Medica of Homeopathic Medicines*. B. Jain Publishers.

APPENDIX-A

PSUEDOCODE

Config.py

```
import os
from dotenv import load_dotenv

# Load environment variables
load_dotenv()

# Azure OpenAI Credentials
AZURE_API_KEY = os.getenv("AZURE_API_KEY")
AZURE_MODEL_NAME = os.getenv("AZURE_MODEL_NAME")
AZURE_API_VERSION = os.getenv("AZURE_API_VERSION")
AZURE_ENDPOINT = os.getenv("AZURE_ENDPOINT")

# Neo4j Credentials
NEO4J_URI = os.getenv("NEO4J_URI")
NEO4J_USERNAME = os.getenv("NEO4J_USERNAME")
NEO4J_PASSWORD = os.getenv("NEO4J_PASSWORD")
NEO4J_DATABASE = os.getenv("NEO4J_DATABASE")
```

Data_extraction.py

```
import os
import glob
from langchain_community.document_loaders import PyPDFLoader

def load_pdfs_from_folder(folder_path: str) -> dict:
  """Load all PDFs from a given folder and its subfolders, and store them in a
dictionary.

  Args:
    folder_path: The path to the folder containing subfolders with PDF files.

  Returns:
```

```
    A dictionary with the subfolder names as keys and the content of all PDFs in each
subfolder as values.
  """
  folder_dict = {}
  j=1
  for subfolder in os.listdir(folder_path):
    subfolder_path = os.path.join(folder_path, subfolder)
    if os.path.isdir(subfolder_path):
      pdf_files = glob.glob(os.path.join(subfolder_path, "*.pdf"))
      documents = []
      print(j, " ", subfolder)
      i = 1
      j+=1
      for pdf_file in pdf_files:
        loader = PyPDFLoader(pdf_file)
        print("  ",i, ".  ", pdf_file)
        documents.extend(loader.load())
        i += 1
      folder_dict[subfolder] = documents
  return folder_dict

folder_path = "/home/vaibhavksir01/Downloads/project/Knowledge Graph/Class 6
textbooks"
raw_document = load_pdfs_from_folder(folder_path)
```

Plot_graph.py

```
from data_extraction import raw_document
from langchain.text_splitter import TokenTextSplitter
from langchain_openai import AzureChatOpenAI
from langchain_neo4j import Neo4jGraph
from langchain_experimental.graph_transformers import LLMGraphTransformer
from config import NEO4J_URI,NEO4J_PASSWORD,NEO4J_USERNAME
from config import AZURE_API_KEY, AZURE_MODEL_NAME,
AZURE_API_VERSION, AZURE_ENDPOINT

text_splitter = TokenTextSplitter(chunk_size = 1024, chunk_overlap = 48)
documents = {key: sorted(text_splitter.split_documents(value), key=lambda doc:
doc.metadata['source']) for key, value in raw_document.items()}

for key, value in raw_document.items():
    for doc in value:
        doc.metadata.clear()
        doc.metadata['producer'] = key

# Initialize Language Model
llm = AzureChatOpenAI(
    azure_deployment=AZURE_MODEL_NAME,
    model=AZURE_MODEL_NAME,
```

```
    api_version=AZURE_API_VERSION,
    azure_endpoint=AZURE_ENDPOINT,
    api_key=AZURE_API_KEY
)

llm_transformer = LLMGraphTransformer(llm = llm) #Transformer that transform
data to graph

#Transforming the Data in a form that can be plotted as a Knowledge graph
graph_document = {}
for key, value in raw_document.items():
    print("start",key)
    graph_document[key]=llm_transformer.convert_to_graph_documents(value)
    print("complete\n")

# Establish connection to Neo4j
graph = Neo4jGraph(url = NEO4J_URI, username = NEO4J_USERNAME,
password = NEO4J_PASSWORD)

#Plotting the Knowledge Graph
for key, value in graph_document.items():
    print(key)
    graph.add_graph_documents(
        value,
        baseEntityLabel=True,
        include_source=True
    )
```

Retrival.py

```
from neo4j import GraphDatabase
from config import NEO4J_URI, NEO4J_USERNAME, NEO4J_PASSWORD,
NEO4J_DATABASE

# Neo4j Connection Class
class Neo4jConnector:
    def __init__(self, uri, user, password, database):
        self._driver = GraphDatabase.driver(uri, auth=(user, password))
        self._database = database
        try:
            # Test connection
            with self._driver.session(database=self._database) as session:
                session.run("RETURN 1")
            print("\nConnected to Graph Database\n")
        except Exception as e:
            print(f"Connection failed: {e}")

    def query(self, cypher_query, parameters=None):
        with self._driver.session(database=self._database) as session:
            result = session.run(cypher_query, parameters)
```

```
            return [record.data() for record in result]

    def close(self):
        self._driver.close()
        print("Connection closed")

# Initialize Neo4j Connection
neo4j_connector = Neo4jConnector(NEO4J_URI, NEO4J_USERNAME,
NEO4J_PASSWORD, NEO4J_DATABASE)

def store_query(word):
    cypher_query = f"""
        MATCH p=(n:Document)-[]->(related)
        WHERE n.text CONTAINS '{word}'
        RETURN related LIMIT 25
        UNION
        MATCH p=(related)-[]->(n:Document)
        WHERE n.text CONTAINS '{word}'
        RETURN related LIMIT 25
        """
    return cypher_query

# Function to Retrieve Data from Neo4j
def retrieve_knowledge(words):
    results = []
    for word in words:
        cypher_query = store_query(word)
        results.extend(neo4j_connector.query(cypher_query))
    return results
```

Agents.py

```
from autogen_agentchat.agents import AssistantAgent
from autogen_ext.models.openai import AzureOpenAIChatCompletionClient
from autogen_agentchat.teams import SelectorGroupChat
from autogen_agentchat.conditions import TextMentionTermination,
MaxMessageTermination
from config import AZURE_API_KEY, AZURE_MODEL_NAME,
AZURE_API_VERSION, AZURE_ENDPOINT

# Initialize Azure OpenAI Client
client = AzureOpenAIChatCompletionClient(
    azure_deployment=AZURE_MODEL_NAME,
    model=AZURE_MODEL_NAME,
    api_version=AZURE_API_VERSION,
    azure_endpoint=AZURE_ENDPOINT,
    api_key=AZURE_API_KEY
)

# ---- Agents ----
```

```
# Planning Agent (Enhanced)
Homeo_PlanningAgent = AssistantAgent(
   "PlanningAgent",
   model_client=client,
   system_message="""
   You are the Planning Agent, the coordinator of a homeopathy diagnosis system.

   Your tasks:
      - Receive the user's symptoms.
      - Send them to the SymptomAnalyzer.
      - If the analysis is incomplete, forward it to the SymptomClarifier.
      - Send the follow-up question back to the user.
      - Once symptoms are sufficient, proceed to RemedyMatcher, then
ExplanationAgent, then ApprovalAgent.
      - If too many clarification attempts fail, exit with a fallback message: "We
recommend consulting a professional."

   Pass task as:
      AgentName : sub-task

   Maintain a logical conversation and stop if 'APPROVED' is received or if
clarification exceeds 2 rounds.
   """
)

# Symptom Analyzer
Homeo_SymptomAnalyzer = AssistantAgent(
   "SymptomAnalyzer",
   model_client=client,
   system_message="""
   You are the Symptom Analyzer.
   Analyze user symptoms to determine if:
      - They're diagnosable.
      - They require clarification.

   If not diagnosable OR partially diagnosable,, output: "Clarification Needed: [List
what's missing]"
   Otherwise, output a clear diagnosis.
   """
)

# Symptom Clarifier
Homeo_SymptomClarifier = AssistantAgent(
   "SymptomClarifier",
   model_client=client,
   system_message="""
   You are the Symptom Clarifier.
   Based on missing or unclear details, craft specific, patient-friendly follow-up
questions.
```

```
    Respond as:
    "Follow-up Question: [your question here]\n\n APPROVED"
    """
)

# Remedy Matcher
Homeo_RemedyMatcher = AssistantAgent(
    "RemedyMatcher",
    model_client=client,
    system_message="""
    You are the Remedy Matcher.
    Based on a diagnosis or clear symptoms, suggest:
        - Remedy name
        - Potency
        - Dosage
        - Instructions

    Keep it aligned with classical homeopathy.
    """
)

# Explanation Agent
Homeo_ExplanationAgent = AssistantAgent(
    "ExplanationAgent",
    model_client=client,
    system_message="""
    You are the Explanation Agent.
    Convert the remedy and diagnosis into a warm, simple explanation for the user.
    Include emotional or lifestyle support tips where applicable.
    Avoid medical jargon.
    """
)

# Approval Agent
Homeo_ApprovalAgent = AssistantAgent(
    "ApprovalAgent",
    model_client=client,
    system_message="""
    You are the final reviewer.
    Tasks:
        - Approve remedy if safe and aligned with principles.
        - Also approve clarification questions if they are logical.

    Output only one word:
    APPROVED
    Or detailed feedback if changes are needed.
    """
)
```

```
# Termination
homeo_termination = TextMentionTermination("APPROVED") |
MaxMessageTermination(15)

# Homeopathy Team Setup
team = SelectorGroupChat(
    [Homeo_PlanningAgent,
     Homeo_SymptomAnalyzer,
     Homeo_SymptomClarifier,
     Homeo_RemedyMatcher,
     Homeo_ExplanationAgent,
     Homeo_ApprovalAgent],
    model_client=client,
    termination_condition=homeo_termination
)

# ---- Optional: Retrieval Agents (if needed for search or future features) ----

# Retrieval Analyzer
Homeo_RetrievalAnalyzer = AssistantAgent(
    "RetrievalAnalyzer",
    model_client=client,
    system_message="""
    You analyze symptom descriptions and extract key elements:
        - Affected area
        - Sensation type
        - Modalities (what improves/worsens)
        - Duration
        - Emotional links

    Output a structured summary of these factors.
    """
)

# Retrieval Agent
Homeo_RetrievalAgent = AssistantAgent(
    "RetrievalAgent",
    model_client=client,
    system_message="""
    You extract relevant keywords from structured symptoms.
    Return only short, search-optimized terms.

    Example: ["burning throat", "better with cold drinks", "morning", "grief"]
    """
)

# Retrieval Team (optional, modular)
retrival_team = SelectorGroupChat(
    [Homeo_RetrievalAgent, Homeo_RetrievalAnalyzer],
    model_client=client,
```

```
    termination_condition=MaxMessageTermination(5)
)
```

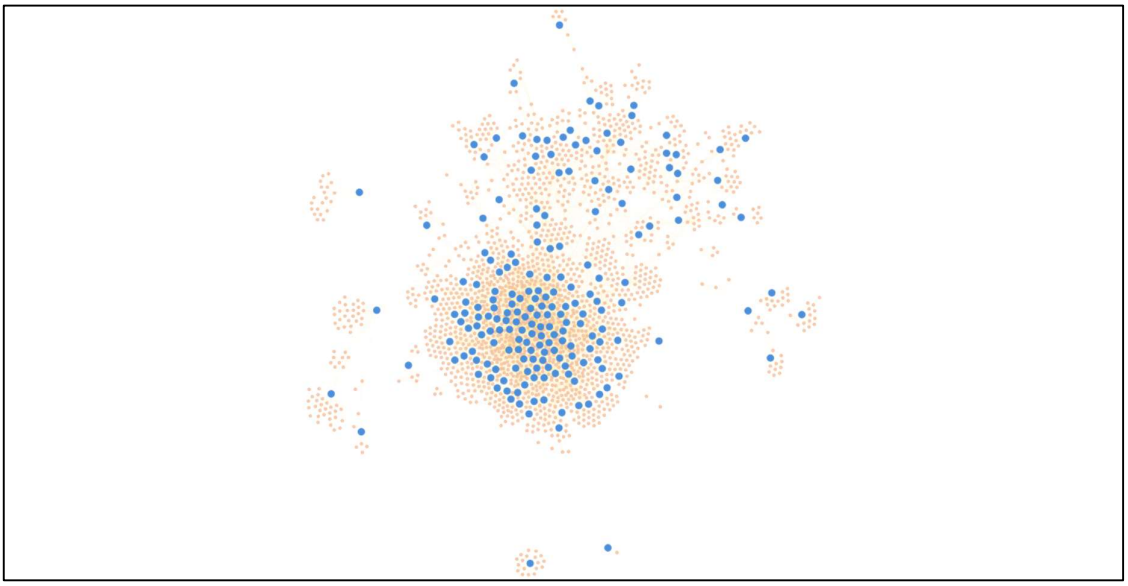
Main.py

```
import asyncio
from agents import team, retrival_team
from symbol import clean_latex_output

fp = open("./file.log", "w")

async def main():
    history = {}
    i=0
    while True:
        i+=1
        user_response = input("\nEnter your question (or type 'exit' to quit): ")
        knowledge = []
        fp = open("./file.log", "w")
        if user_response.lower() == 'exit':
            print("\nExiting...\n")
            break

        retrival_task = f"symptoms:{user_response},history:{history}"
        async for result in retrival_team.run_stream(task=retrival_task):
            retrival_response = []
            if hasattr(result, 'messages') and hasattr(result.messages[-2], 'content'):
                response = str(result.messages[-2].content).replace("Retrival:",
"").replace("","").replace("\n","").replace("[","").replace("]","").replace("Retrieval:",
"").replace("","").replace(",","").replace(r':\\',')
                retrival_response.extend(response.strip(",").split())
                fp.write(f"list with Duplicate Elements: \n{retrival_response}\n")
                for i in range(len(retrival_response)):
                    x=''
                    for w in retrival_response[i]:
                        if (str(w).isalpha() or str(w).isnumeric()) and w!='.':
                            x+=w
                        else:
                            x+=" "
                    if ~x.isspace():
                        retrival_response[i]=x
                retrival_response = list(set(retrival_response))
                fp.write(f"\nWords retrived:\n{retrival_response}\n")

        from retrieval import retrieve_knowledge
        knowledge.extend(retrieve_knowledge(retrival_response))

        if not knowledge:
            fp.write("No relevant knowledge found in Neo4j.\n")
            continue
```

```
        fp.write(f"\nRetrieved knowledge:\n {knowledge}\n")

        task = f"""question: {user_response}
        context: {knowledge}
        history: {history}
"""

        async for result in team.run_stream(task=task):
            fp.write(f"\nAgent Output\n{result}\n")
            if hasattr(result, 'messages'):
                for i in range(-2,-(len(result.messages)+1),-1):
                    if 'user' not in result.messages[i].content.lower().split():
                        response = result.messages[i].content
                        response = clean_latex_output(response)
                        print(f"\nChatbot:\n{response}\n")
                        history[i]={"symptom":user_response,"diagosis":response}
                        break
                    else:
                        continue

if __name__ == "__main__":
    asyncio.run(main())
```

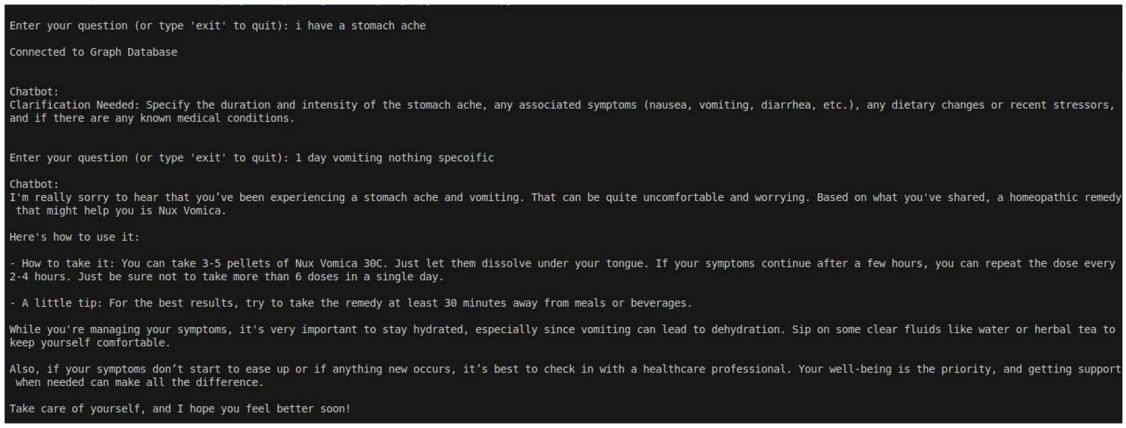
APPENDIX-B

SCREENSHOTS



```
Enter your question (or type 'exit' to quit): i have a headache

Connected to Graph Database


Chatbot:
Clarification Needed:
- Duration and frequency of the headache
- Location and nature of the pain (e.g. throbbing, constant)
- Any accompanying symptoms (e.g. nausea, sensitivity to light)
- Potential triggers (e.g. stress, food, changes in sleep)
- Previous medical history related to headaches
- Any medications currently taken or previously tried for relief


Enter your question (or type 'exit' to quit): i have been having it for 2 days and its continuous in forehead no additional symptoms i am suffuring from thyroid

Chatbot:
It sounds like you've been dealing with a stubborn headache for the past couple of days, and that can be really tough! It's good to know you want to find some relief. Based on you
r symptoms—a continuous headache located in your forehead—I've got a gentle homeopathic remedy that might help.

Remedy Name: Bryonia Alba
Potency: 30C
Dosage: Take 2 pellets three times a day.

How to Take It:
1. Let the pellets dissolve slowly under your tongue.
2. Try to take them away from food and strong tastes (like coffee or mint) for about 30 minutes before and after.

Keep an eye on how you feel. If your headache doesn't get better after 48 hours, it might be a good idea to reach out to a homeopathic practitioner for more support.

In the meantime, remember to take care of yourself. Getting enough rest, staying hydrated, and possibly finding some quiet time can really help soothe your head and your spirit. A
nd since you mentioned thyroid issues, keeping stress levels low is important, too—maybe consider some gentle stretches or relaxation techniques. You're not alone in this, and I h
ope you feel better soon!
```

```
Enter your question (or type 'exit' to quit): exit

Exiting...
```

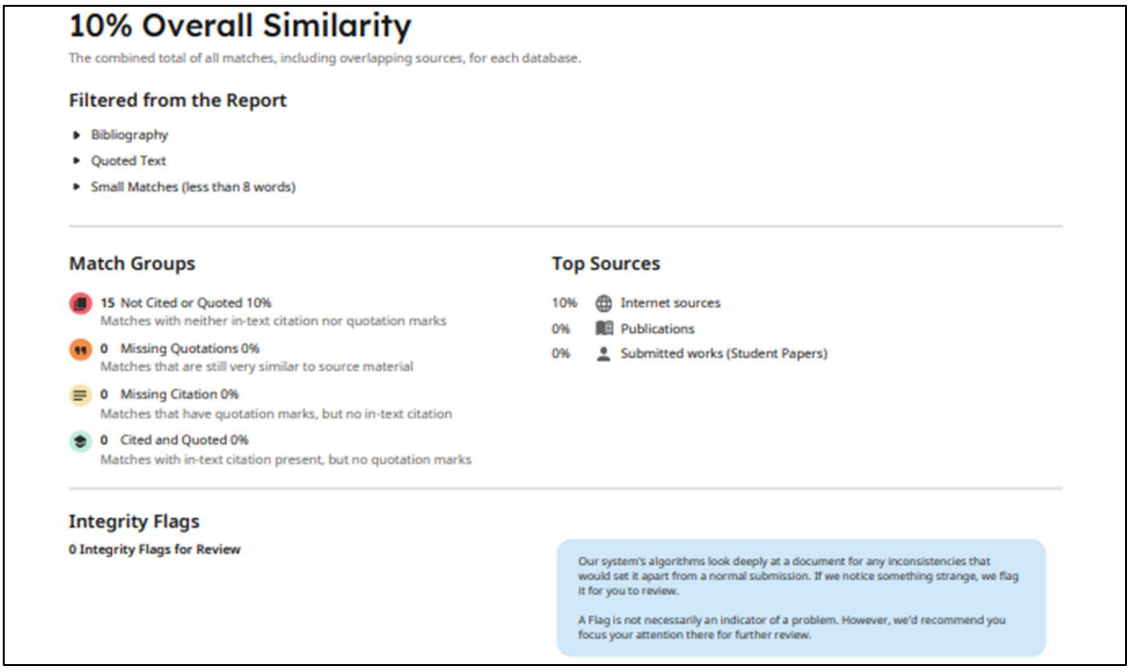

Enter your question (or type 'exit' to quit): i have a stomach ache
Connected to Graph Database
Chatbot:
Clarification Needed: Specify the duration and intensity of the stomach ache, any associated symptoms (nausea, vomiting, diarrhea, etc.), any dietary changes or recent stressors, and if there are any known medical conditions.
Enter your question (or type 'exit' to quit): 1 day vomiting nothing specoific
Chatbot:
I'm really sorry to hear that you've been experiencing a stomach ache and vomiting. That can be quite uncomfortable and worrying. Based on what you've shared, a homeopathic remedy that might help you is Nux Vomica.
Here's how to use it:
- How to take it: You can take 3-5 pellets of Nux Vomica 30C. Just let them dissolve under your tongue. If your symptoms continue after a few hours, you can repeat the dose every 2-4 hours. Just be sure not to take more than 6 doses in a single day.
- A little tip: For the best results, try to take the remedy at least 30 minutes away from meals or beverages.
While you're managing your symptoms, it's very important to stay hydrated, especially since vomiting can lead to dehydration. Sip on some clear fluids like water or herbal tea to keep yourself comfortable.
Also, if your symptoms don't start to ease up or if anything new occurs, it's best to check in with a healthcare professional. Your well-being is the priority, and getting support when needed can make all the difference.
Take care of yourself, and I hope you feel better soon!

APPENDIX-C

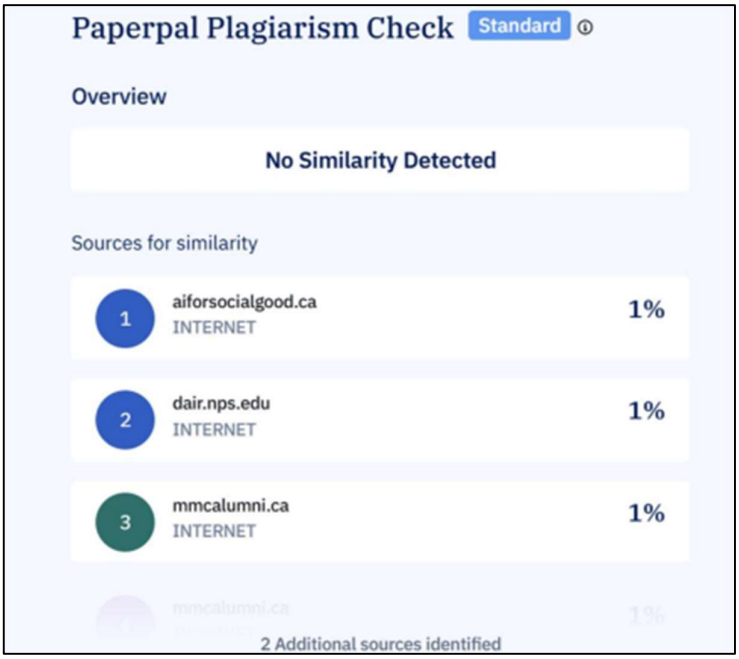
ENCLOSURES

1.Similarity Index / Plagiarism Check report clearly showing the Percentage

Project Report



Research Paper



3. Details of mapping the project with the Sustainable Development Goals (SDGs).

1**. SDG 3: Good Health and Well-Being**

- The project enhances access to alternative healthcare through AI-powered homeopathy advice, offering users personalized, non-invasive remedies and guidance.
- By making homeopathy knowledge more accessible, it helps individuals manage their health proactively, contributing to overall well-being and supporting sustainable health practices.

2. **SDG 9: Industry, Innovation, and Infrastructure**

- The project leverages cutting-edge technologies like AI, Natural Language Processing (NLP), and machine learning to innovate in the field of homeopathy, creating an intelligent, scalable solution for personalized health management.
- It supports the development of resilient digital infrastructures that can handle growing datasets, ensuring the project's capacity to scale effectively and deliver reliable healthcare information to users worldwide.