

THE FUTURE OF AYURVEDA: HARNESSING THE POWER OF ARTIFICIAL INTELLIGENCE FOR PERSONALIZED TREATMENT AND DIAGNOSIS

GROUP REPORT

B.Sc. (Hons) Degree in Information Technology Specialized in Software
Engineering





Department of Computer Science and Software Engineering

Sri Lanka Institute of Information Technology Sri Lanka

September 2023

DECLARATION

I declare that this is my own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
Maduwantha K.A.I	IT20069186	
De Silva A. S	IT20166038	
Senarathne S M A D	IT20089436	
Jayasinghe J A S C	IT20216078	

Signature of the Supervisor

Date

(Dr. Darshana Kasthurirathna)

03/11/2023

.....

.....

ABSTRACT

The rapid and busy nature of contemporary living often results in an uneven distribution of daily activities, inadequate dietary habits, insufficient physical exercise and leisure time, and excessive work-related stress, which can lead to poor health and dissatisfaction. While Ayurveda provides alternative solutions for many diseases. But people may have difficulties identifying the appropriate herbs and treatments and consulting with doctors in a timely and cost-effective manner. Additionally, the high cost of Western medicine may be a barrier, and not all illnesses are treatable. The proposed solution aims to assist users in discovering interactive Ayurvedic-based treatments for various symptoms. This solution is anticipated to be beneficial for those seeking alternatives to conventional medicine.

The identification of Ayurvedic medical herbs is a critical step in the treatment of various diseases. Traditional methods of identification involve physical examination and experience, which can be time-consuming and prone to errors. To address this issue, this study proposes a new approach that uses image processing and machine learning techniques to identify Ayurvedic medical herbs for the treatment of diseases such as diabetes, arthritis, and asthma.

The study then collected and processed large amounts of image data from various geographical areas to map the distribution of herbal plants. Finally, the model was implemented in a software application for practical use. As a novelty improvement, the study incorporated continual learning/transfer learning to improve accuracy and auto-machine learning to make it easier to add new plants. The proposed approach has the potential to significantly improve the accuracy and efficiency of Ayurvedic medical herb identification and treatment, making it a valuable tool for the healthcare industry.

TABLE OF CONTENTS

DECLARATION.....	2
ABSTRACT.....	3
TABLE OF CONTENTS	4
ACKNOWLEDGEMENT	6
LIST OF FIGURES	7
LIST OF TABLES	9
LIST OF ABBREVIATIONS	9
LIST OF APPENDICES	10
1 INTRODUCTION	11
1.1 Background & Literature Survey	11
1.2 Research Gap.....	23
2 RESEARCH OBJECTIVES.....	28
2.1 Main Objectives	28
2.2 Specific Objectives.....	30
3 METHODOLOGY	34
3.1 Requirement gathering and Analysis.	34
3.2 Feasibility studies.....	35
3.3 System Diagrams.....	37
3.3.1 Overall system diagram	37
3.3.2 Individual component diagrams.....	38
3.4 Understanding the Key Pillars of the Research Domain.....	39
3.4.1 Machine learning	39
3.4.2 Natural Language Processing (NLP)	41
3.4.3 Convolutional Neural Network (CNN).....	43
3.4.4 Data augmentation	46
3.5 Specific Methodology	47
3.6 Commercialization aspects of the system	50
3.7 Consideration of the aspect of the system.....	50

3.7.1	Social aspects	50
3.7.2	Security aspects.....	51
3.7.3	Ethical aspects.....	51
4	IMPLEMENTATION AND TESTING	51
4.1	Implementation.....	51
4.1.1	Model implementation.....	54
4.2	Testing	73
4.2.1	Test plan and test strategy	73
5	RESULTS AND DISCUSSIONS	73
5.1	Results	73
5.1.1	For the chatbot	73
5.1.2	For the herbal plant identification:.....	74
5.1.3	For the doctor recommendation system:.....	75
5.1.4	For the social network:.....	78
5.2	Discussion	79
5.2.1	The integration of the Rasa chatbot architecture with the BERT	79
5.2.2	Enhancing Ayurvedic healthcare through image identification	79
5.2.3	Enhancing Ayurvedic healthcare through doctor recommendation systems.....	81
5.2.4	Social network	82
6	CONCLUSION	83
7	REFERENCES	84
8	APPENDIX	86
8.1	Turnitin Report.....	86

ACKNOWLEDGEMENT

First and foremost, we want to sincerely thank our mentor Dr. Darshana Kasthurirathna for his unwavering support and direction, which enabled us to successfully complete my undergraduate research. Along with our supervisor, Dr. Samantha Rajapaksha, the co-supervisor of this research project deserves my deepest gratitude for always being available to assist. Due to the fact that this research project combines technology and ayurveda, both technology specialists and ayurvedic professionals were needed for advice and support. It is very appreciated that Dr. Mrs. Janaki Bandara who provided such extensive help throughout the project to close the knowledge gap in those fields. My sincere gratitude also goes out to the other doctors of Gampaha Wikramarachchi Ayurvedic University. Last but not least, I would like to convey our thanks to everyone who has helped with this project in some way, whether directly or indirectly, including my teammates, family, and friends.

LIST OF FIGURES

Figure 1:Survey on idea of people about ayurvedic medicine.....	13
Figure 2:Survey on having ayurvedic treatments before.	13
Figure 3:Survey on the frequency of home based ayurvedic treatments usage	14
Figure 4:Survey in Sinhala on the frequency of home based ayurvedic treatments usage	14
Figure 5:Survey on the preference recommendations from a chatbot over a human doctor.	15
Figure 6:Survey in Sinhala on the preference recommendations from a chatbot over a human doctor.	16
Figure 7:Survey on the how comfortable are people sharing personal health information with a chatbot.....	16
Figure 8:Survey on how frequently people use home based ayurvedic treatments.....	17
Figure 9:Survey in Sinhala on the frequency of home based ayurvedic treatments usage	17
Figure 10:Survey in what social platforms do you use?	19
Figure 11:Survey in Sinhala what social platforms do you use?	19
Figure 12:Survey in where you find a specilaist.....	20
Figure 13:Survey in Sinhala where you find a specilaist.....	20
Figure 14:Overall System Architecture Diagram	37
Figure 16: Individual component diagram 1	39
Figure 15:: Individual component diagram 2.....	38
Figure 17:: Individual component diagram 4.....	39

Figure 18: CNN Architecture.....	44
Figure 19:Process of Model Training	45
Figure 20:Data Augmentation.....	47
Figure 21:Code Snippet of BERT Model	54
Figure 22:Code Snippet of training loop	55
Figure 23:Code Snippet of training loop	57
Figure 24:Code Snippet to splitting data into training and test data.....	59
Figure 25:Code Snippet to Data Processing	61
Figure 26:Code Snippet to model building.....	62
Figure 27:ode Snippet to image classification model using deep learning.....	65
Figure 28:Model Implementation code snippet	66
Figure 29::Sentiment Score Analysis code snippet	67
Figure 30:Distance Sorter Algorithm.....	68
Figure 31:Rating and Reviews Sorter Algorithm code snippet	69
Figure 32:Process Singular Form of a Word	71
Figure 33:Process Co-Reference Resolution	72
Figure 34: Confusion Matrix	75
Figure 35: Classification Rreport.....	76
Figure 36: Turnitin Report	86

LIST OF TABLES

Table 1: List of Abbreviations	10
Table 2: Research Gap Summary 1	24
Table 3: Research Gap Summary 2	26
Table 4: Research Gap Summary 3	27
Table 5: CNN Comparisons.....	74

LIST OF ABBREVIATIONS

Abbreviation	Description
AI	Artificial Intelligence
ML	Machine Learning
EHR	Electronic Health Records
AML	Auto Machine Learning
CNN	Convolutional Neural Network
SVM	Support Vector Machine
SDLC	Software Development Life Cycle
HOG	Histogram of Oriented Gradients
SIFT	Scale-Invariant Feature Transform

TP	True Positives
FP	False Positives
TN	True Negative
FN	False Negative
YOLO	You Only Look Once

Table 1: List of Abbreviations

LIST OF APPENDICES

Appendix A: Turnitin Report

1 INTRODUCTION

1.1 Background & Literature Survey

Ayurveda is a traditional medical practice with Indian roots that places a strong emphasis on a holistic view of health and wellness. It uses organic treatments, dietary adjustments, and individualized care based on a person's dosha, or body type. Because of its efficacy and focus on prevention and individualized care, Ayurveda has become increasingly well-known on a global scale. A more specialized and individualized approach to treatment is needed for non-communicable diseases like diabetes, heart disease, and cancer because of the fast-paced modern lifestyle. In order to provide individualized and efficient therapy, this has led researchers and practitioners to investigate how Ayurveda principles might be combined with cutting-edge technology like Artificial Intelligence (AI).

Our research indicates that Sri Lankans have a long-standing cultural and traditional connection to Ayurveda. The history of the nation dates back more than 3,000 years, and Sri Lankan culture is firmly rooted in the practice of ayurveda. In Sri Lanka, different ailments have been treated and prevented through Ayurveda practices for thousands of years. The popularity of Ayurveda in Sri Lanka can be ascribed to its all-encompassing method of healthcare, which places an emphasis on the harmony of the mind, body, and spirit. The natural and organic aspects of Ayurveda, which uses herbs and other natural therapies to address a variety of health conditions, have long been valued by Sri Lankans. For minor ailments and long-term health concerns, Sri Lankans favor Ayurveda therapies over contemporary medicine.

Ayurveda, an ancient Indian healthcare system, prioritizes holistic well-being, natural remedies, and personalized treatment based on one's dosha. Its global popularity stems from its effectiveness and prevention-focused approach. To meet the demands of modern life and non-communicable diseases, Ayurveda is integrating with Artificial Intelligence (AI) for personalized care.

In Sri Lanka, Ayurveda has a rich history of over 3,000 years, deeply ingrained in the culture. People appreciate its holistic approach, natural remedies, and favor it over modern medicine, especially for minor ailments. The country boasts Ayurvedic hospitals and trained doctors who

treat a wide range of conditions. Recent interest and government support are fueling Ayurveda's resurgence.

AI is revolutionizing healthcare by analyzing vast patient data for diagnosis, treatment, and drug discovery. It aids radiologists in medical image analysis and helps create personalized treatment plans based on patient histories. AI chatbots and virtual assistants offer initial diagnosis and remote monitoring, enhancing efficiency and personalization in healthcare. Surveys indicate that many Sri Lankans are open to Ayurvedic treatment recommendations from AI chatbots. However, preferences vary, highlighting the importance of individual choice in AI technology adoption. AI's potential in Ayurveda includes personalized treatments, improved diagnosis, and increased patient engagement.

Several Ayurvedic apps, such as AyurMana and Wealthy, already utilize AI for personalized treatment recommendations. These apps cater to individuals seeking natural remedies and personalized healthcare. Proposed innovations include a conversational AI chatbot for personalized solutions and advice for specific symptoms. An image processing component identifies needed herbal plants and connects users with Ayurvedic doctors. A social network allows users to discuss health-related knowledge and contribute to a community knowledge base. AI is transforming healthcare by analyzing vast patient data for diagnosis and treatment. Radiologists benefit from AI algorithms in medical image analysis, while AI aids in creating personalized treatment plans based on patient history. AI-powered apps like AyurMana and Wealthy offer personalized Ayurvedic recommendations.

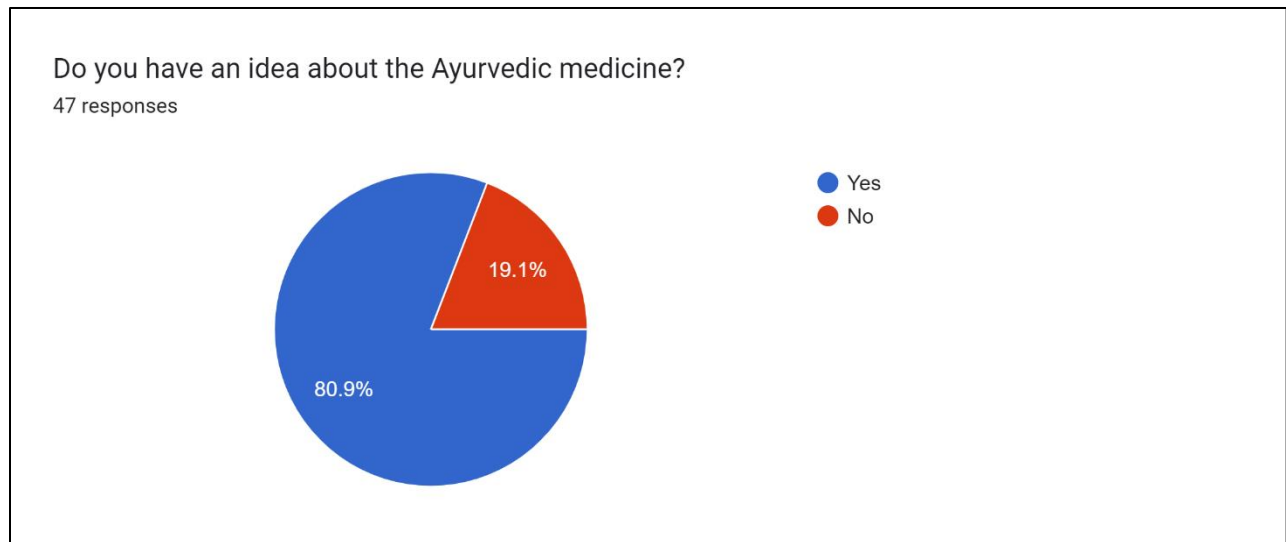


Figure 1: Survey on idea of people about ayurvedic medicine

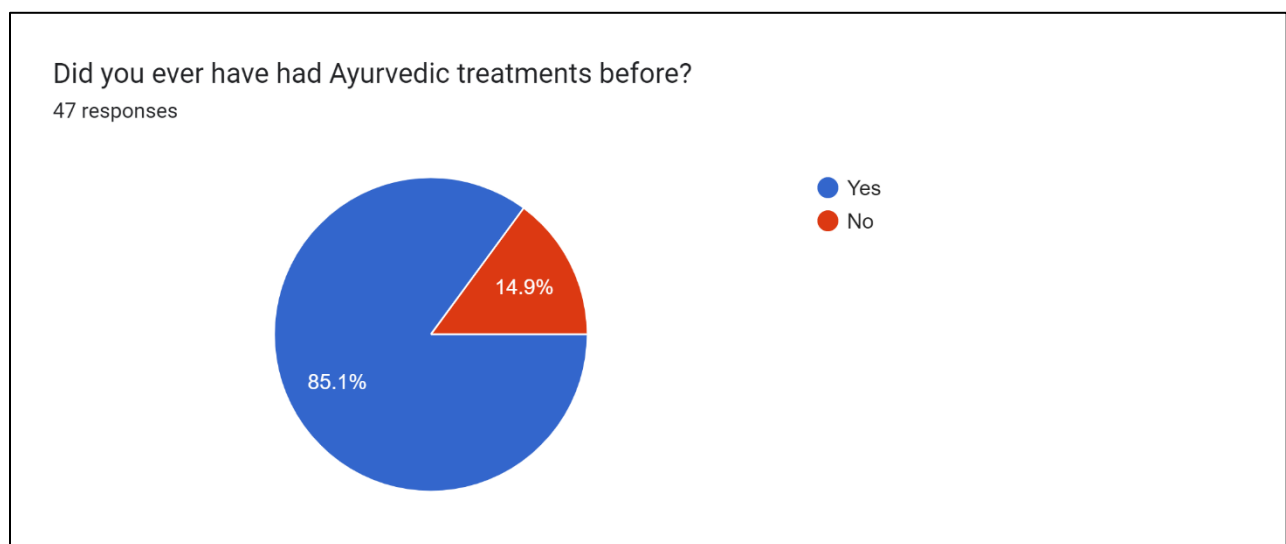


Figure 2: Survey on having ayurvedic treatments before.

From the above two survey results it shows that most of the people in Sri Lanka are aware about the ayurvedic medicine and most people had used them even once in their lifetime. Considering the percentages more than 80% of people are aware of ayurvedic medicine.

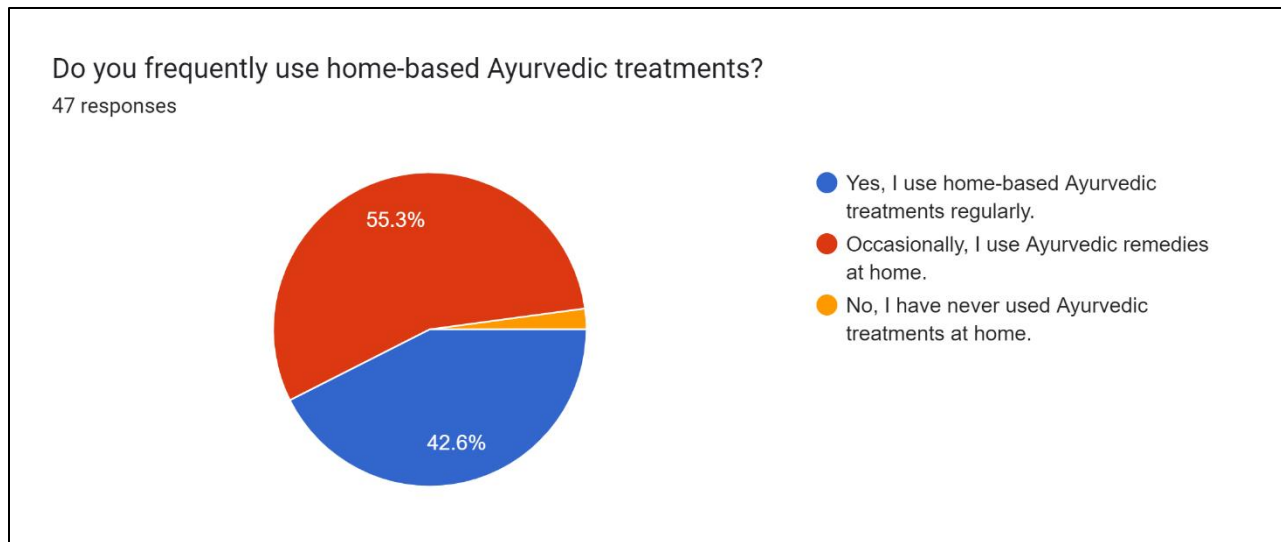


Figure 3: Survey on the frequency of home based ayurvedic treatments usage

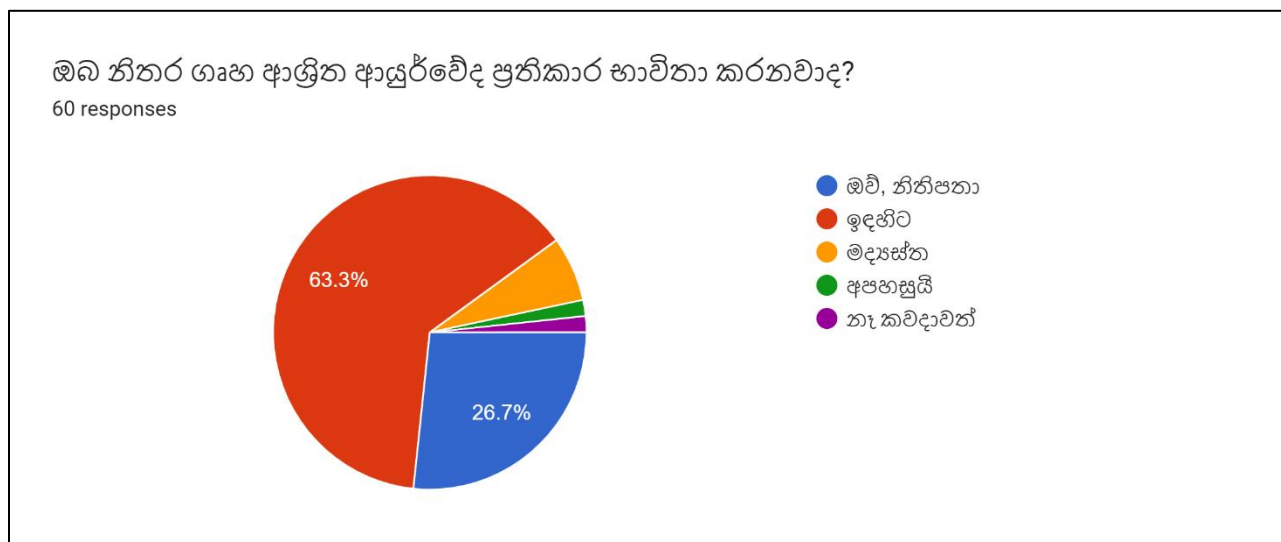


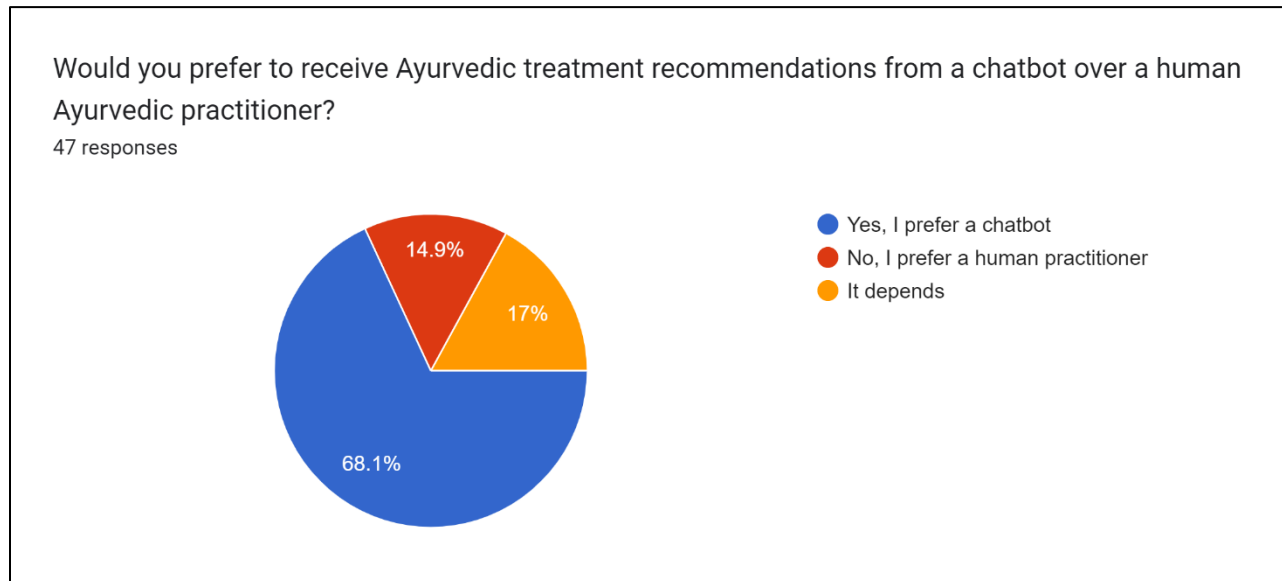
Figure 4: Survey in Sinhala on the frequency of home based ayurvedic treatments usage

Based on the survey results, it can be concluded that more than half of the respondents are frequently using ayurvedic treatments.

Overall, our research indicates that Sri Lankan people have a deep appreciation for Ayurveda and its holistic approach to healthcare. The popularity of Ayurveda in Sri Lanka is expected to continue to grow as people seek out natural and organic alternatives to modern medicine.

AI is transforming healthcare with innovative solutions for diagnosis, treatment, and monitoring. AI has the ability to accurately and efficiently analyze large amounts of patient data, making it incredibly useful in healthcare. AI algorithms can identify patterns in medical images such as CT scans, MRIs, and X-rays to assist radiologists in identifying potential abnormalities or diseases.

Additionally, AI can help doctors create personalized treatment plans by analyzing data on



treatment outcomes and patient health records. AI can also aid in the discovery of new drugs and treatments by analyzing large datasets to identify patterns and provide insights. Furthermore, AI-powered chatbots and voice assistants can provide initial diagnosis and recommend next steps. AI can be used for remote patient monitoring, alerting doctors to potential issues, analyzing electronic health records, and providing personalized health advice through virtual assistants. Overall, AI has enormous potential to improve patient outcomes and provide personalized healthcare solutions.

Figure 5: Survey on the preference recommendations from a chatbot over a human doctor.

According to the survey results more than half of respondents are preferring to have ayurvedic treatment recommendations from a chat bot over a human doctor.

වෛද්‍යවරයෙකුට වඩා චැට්බෝට් එකකින් ආයුර්වේද ප්‍රතිකාර නිර්දේශ ලබා ගැනීමට ඔබ කැමතිද?
60 responses

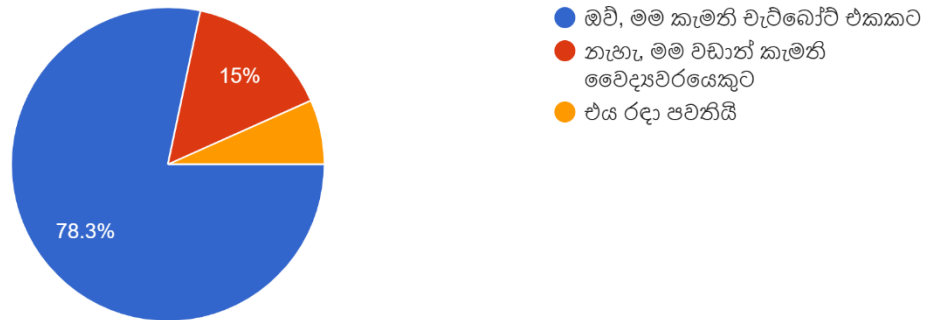


Figure 6: Survey in Sinhala on the preference recommendations from a chatbot over a human doctor.

The results suggest that there are no clear consequences on whether respondents would prefer to receive Ayurvedic treatment recommendations from a chatbot or a human practitioner. It highlights the importance of considering the preferences and concerns of individuals when implementing AI technology in healthcare.

How comfortable are you sharing personal health information with a chatbot to receive personalized Ayurvedic treatment recommendations?
47 responses

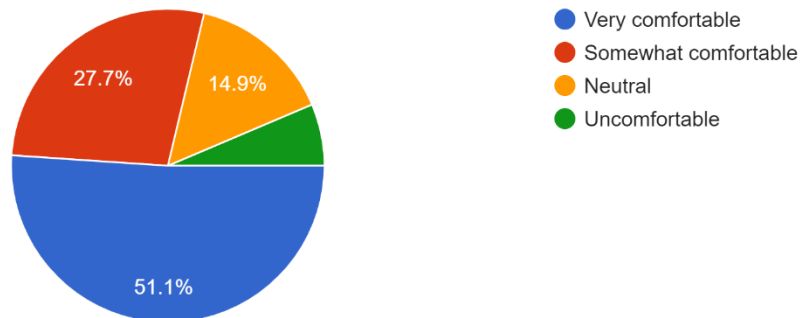


Figure 7: Survey on the how comfortable are people sharing personal health information with a chatbot.

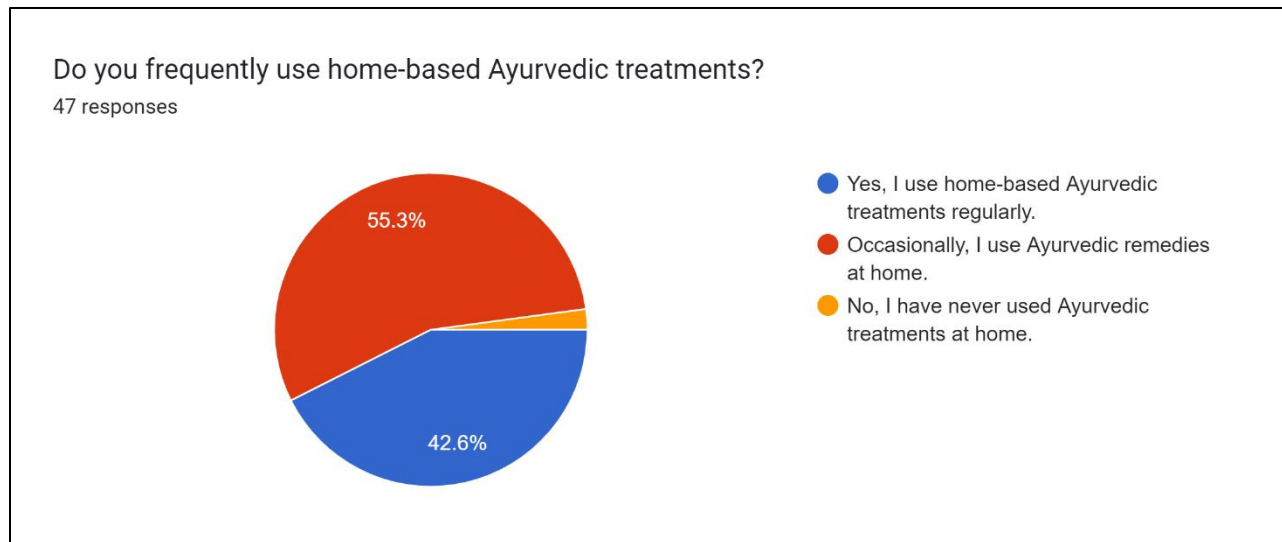


Figure 8: Survey on how frequently people use home based ayurvedic treatments.

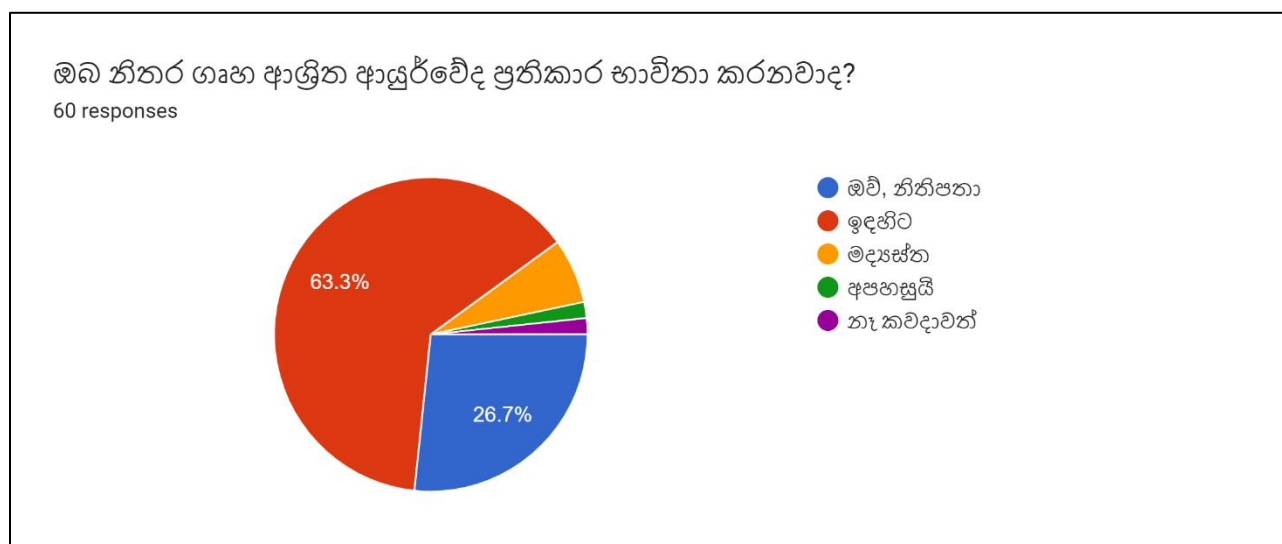


Figure 9: Survey in Sinhala on the frequency of home based ayurvedic treatments usage

Based on the survey results, it can be concluded that more than half of the respondents are frequently using ayurvedic treatments.

Ayurvedic hospitals and clinics may be found all over Sri Lanka, and Ayurveda has a significant influence on the country's healthcare system. The people of Sri Lanka have access to skilled and knowledgeable Ayurveda medical professionals who offer their patients individualized care and

therapies. These medical professionals treat a variety of illnesses, from the ordinary cold to more serious disorders like cancer and diabetes, using Ayurveda treatments and conventional healing methods. As people in Sri Lanka become more health-conscious and explore for natural and organic alternatives to modern medicine, there has been a resurgence in interest in Ayurveda recently. Ayurveda is a crucial component of Sri Lanka's healthcare system, and the government has taken steps to promote and advance it.

The integration of AI in Ayurveda has the potential to revolutionize personalized healthcare. It allows for tailored treatments, improved disease diagnosis, and increased patient engagement. Ayurveda can leverage various AI technologies, including machine learning, natural language processing, and computer vision. Several Ayurvedic and plant identification apps are already available, catering to individuals seeking natural remedies and personalized healthcare. Surveys indicate that most Sri Lankans are willing to join an Ayurvedic online platform, highlighting the growing interest in Ayurveda. Proposed innovations include a conversational AI chatbot for personalized solutions and an image processing component for herbal plant identification. A social network will allow users to share health-related knowledge, contributing to a community knowledge base.

To ensure accuracy, expertise in Ayurveda is essential. Data will be gathered from social network communities and Ayurvedic doctors' details for supervised machine learning algorithms. In conclusion, the integration of AI in Ayurveda holds great promise for personalized healthcare. As non-communicable diseases become more prevalent, the synergy between AI and traditional Ayurveda can lead to improved health outcomes and a brighter future for healthcare.

Therefore, AI integration in Ayurveda holds promise for personalized healthcare, better diagnosis, and treatment outcomes. The synergy between traditional Ayurveda and AI technology can enhance health and wellness in a rapidly changing world.

In today's digital age, the healthcare-seeking journey has been significantly influenced by social media and online resources. Survey data highlights the prevalence of Facebook groups (80.4%) as a preferred platform for healthcare information due to its vast user base and diverse healthcare-

related groups. In contrast, platforms like Quora, Discord, Telegram, and Medical Sciences Stack Exchange have lower utilization, possibly due to perceived limitations or niche focus.

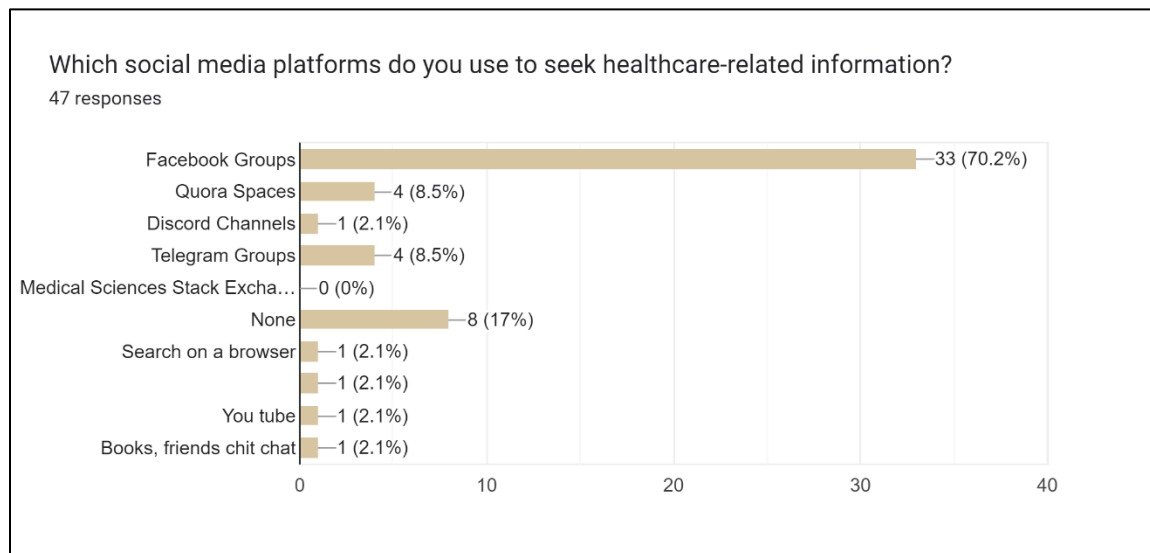


Figure 10: Survey in what social platforms do you use?

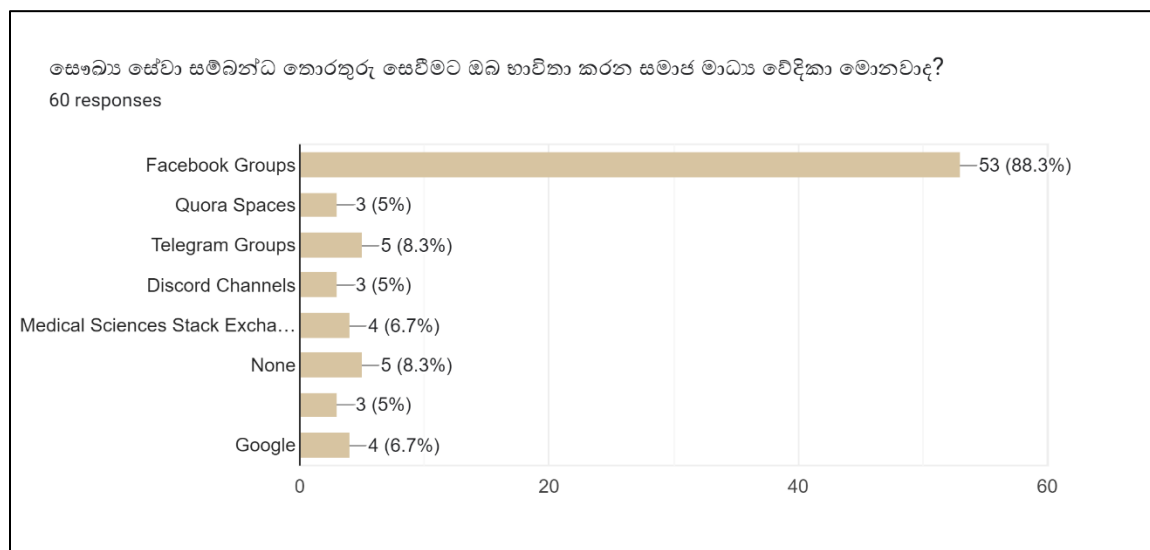


Figure 11: Survey in Sinhala what social platforms do you use?

A noteworthy segment (11.2%) abstains from using any social media platform for healthcare information, possibly due to privacy concerns or doubts about online information credibility.

Language and cultural factors play a significant role in healthcare information-seeking behavior, as illustrated by Figure 1 and Figure 2.

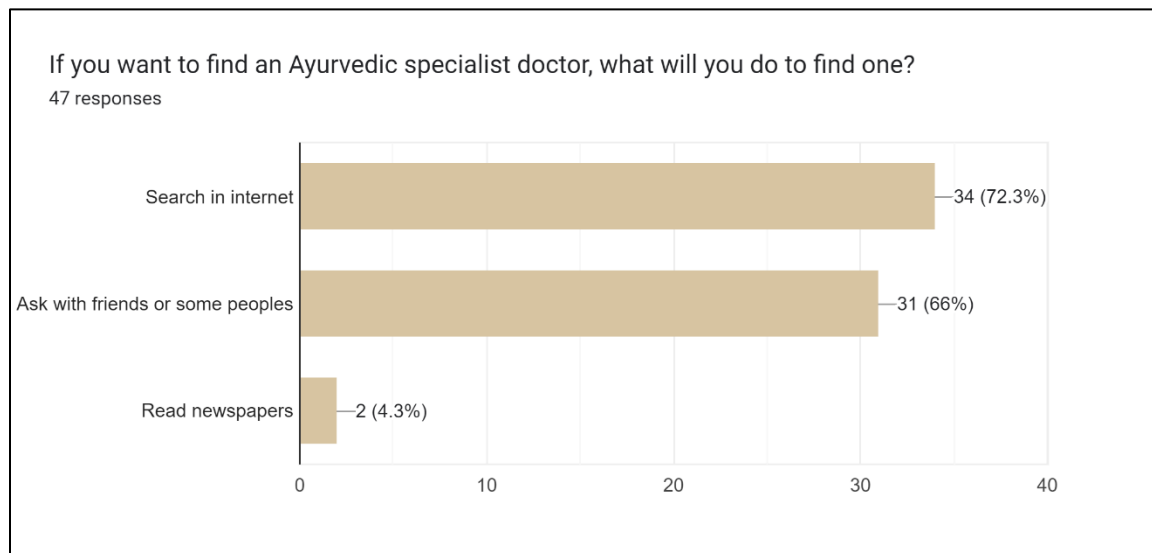


Figure 12: Survey in where you find a specialist.

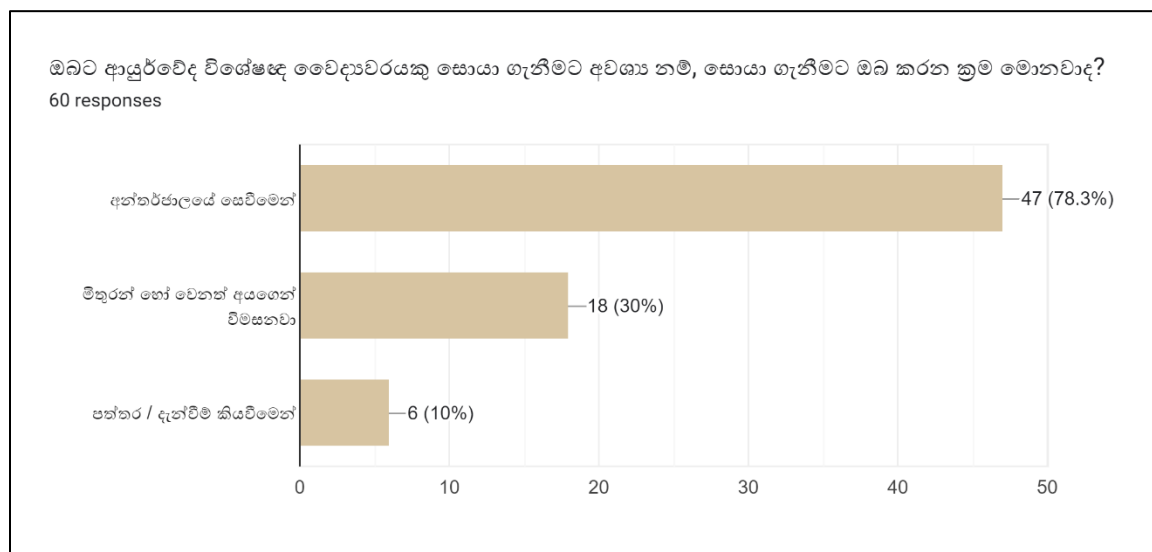


Figure 13: Survey in Sinhala where you find a specialist.

Regarding Ayurvedic medicine, the internet is the primary source (75.2%) for finding Ayurvedic specialists, highlighting the digital platform's role in accessing alternative healthcare options. Personal networks (48.6%) remain influential, emphasizing the power of word-of-mouth referrals

in healthcare decisions. Some individuals (7.9%) still rely on traditional media like newspapers for Ayurvedic specialist information, suggesting opportunities for traditional-digital collaboration.

Social media analytics offer insights into the credibility, uniqueness, and frequency of information across social media [21]. Credibility is especially crucial for health-related content, as it can have life-altering consequences. Social media can be helpful and harmful in this context, so it's important to carefully evaluate health-related content by examining the source, evidence, accuracy, tone, and target audience.

When evaluating health information on social media, it's important to check the author's credentials and expertise, rely on scientific evidence, look for transparency and referencing, check for accuracy and consistency, and be aware of any bias or agenda [21]. By doing so, individuals can make informed decisions about the credibility of the information they encounter.

When checking health information on social media, it's important to consider recency. Look at the publication date, how often it's updated, and if it's relevant to current health practices and guidelines. Check the reputation of cited sources and if it's validated by other authoritative sources or health professionals. This process helps ensure the information's accuracy, reliability, and contemporaneity.

When evaluating health content on social media, unique information requires careful assessment. Check the source's credibility, supporting evidence, consistency with established sources, and potential for bias. Seek additional sources and perspectives to make well-informed health decisions.

When evaluating information on social media, consider the frequency of postings, but don't forget to assess the source's credibility and supporting evidence [21]. Look for consistency with authoritative sources, potential bias, and relevance to current health practices. With so much information available, it's important to cross-reference and seek alternative perspectives for well-informed decisions.

When evaluating health information on social media, it's important to consider its relevance, quality, and potential bias. Seek information from authoritative sources and consult healthcare

professionals for credibility and relevance to your unique health needs. Going deeper into natural language processing can help understand salience but be mindful of privacy concerns.

To assess health information on social media, consider credibility, recency, uniqueness, frequency, and salience. This helps make informed decisions about health and well-being. But, be cautious of privacy concerns when using advanced natural language processing techniques to analyze salience.

1.2 Research Gap

The integration of AI technology into Ayurvedic medicine is still in its early stages, with significant research gaps. One key area lacking development is personalized treatment through conversational AI chatbots for non-communicable diseases in Ayurveda. Existing mobile apps offer basic solutions but lack the sophistication that AI can provide. The absence of standardization in Ayurvedic medicine complicates the development of effective algorithms for data analysis.

Another research gap is the application of AI in the manufacturing and quality control of Ayurvedic products, given the lack of standardization in the field. AI can enhance the safety and consistency of Ayurvedic products by analyzing ingredient data and manufacturing processes.

Research highlights the potential of medical chatbots for personalized medicine, emphasizing user-friendliness and accessibility. The study suggests improving symptom recognition capabilities by incorporating factors like location and symptom intensity. Some Research presents a medical chatbot as a diagnostic assistant that can replace conventional diagnosis methods, emphasizing accurate symptom analysis and personalized recommendations. One research was there which focuses on virtual doctors through chatbots for health management, underlining their role in catering to individuals with time constraints and the convenience of diagnosing diseases with a single click. Another one describes the automation of a medical chatbot for personalized recognition, using an external recognition engine but hinting at the need for an in-house engine.

While these studies highlight the potential of medical chatbots in disease diagnosis and treatment recommendations, future work should refine symptom recognition algorithms and optimize chatbot deployment for enhanced performance.

Features	Existing Mobile Applications	Proposed System
Ayurvedic Chatbot	Yes	Yes
Availability of personalized solutions	Yes	Yes

User Friendly Platform	No	Yes
Availability of a knowledge base	No	Yes
Chatbot framework	Yes	Yes
Ability to ask to follow back questions	Yes	Yes
Using RASA	No	Ye
Updating Knowledgebase Easily	No	Yes

Table 2: Research gap summary 1

The integration of AI technology for identifying Ayurvedic medical herbs needed for non-communicable disease treatments using image processing and machine learning is an emerging field. Existing image recognition models are not optimized for Ayurvedic herbs and lack specificity for disease-related treatments. There is also a scarcity of research on mapping the locations of these herbs, crucial for sustainable cultivation. Research A discusses the development of plant identification technologies based on image processing and pattern recognition. While it offers

insights into automatic plant recognition, it does not focus on Ayurvedic herbs or disease-related treatments.

One research demonstrates the use of machine learning techniques like ANN and SVM for identifying Ficus species based on leaf photos. However, it is specific to Ficus species and not Ayurvedic herbs. Another one research evaluates automatic plant identification using deep neural networks and large datasets but does not target Ayurvedic herbs or disease treatments. The proposed system aims to bridge these gaps by specifically focusing on Ayurvedic medical herbs for non-communicable disease treatments using image processing and machine learning. It will also incorporate geographic mapping to locate these herbs, addressing challenges such as variable plant morphology and environmental conditions. In summary, while existing research provides valuable insights into plant identification and classification using image processing and machine learning, there is a clear need for dedicated research in the context of Ayurvedic herbs and their applications in disease treatments.

Features	Existing Mobile Applications	Proposed System
Ayurvedic Plants Identification	Yes	Yes
Usage of image processing	Yes	Yes
Plant Identification	Yes	Yes
Representative datasets	Yes	Yes

Noise sensitivity	No	Yes
Using of auto machine Learning	No	Yes

Table 3: Research gap summary 12

The healthcare industry is witnessing a digital transformation, with a growing recognition of the potential of artificial intelligence (AI) in Ayurvedic medicine. However, there is a significant research gap in the development of AI-powered mobile applications to help patients choose suitable Ayurvedic doctors based on their specific symptoms and needs. While there is existing research on AI's application in diagnosis and treatment recommendations in Ayurvedic medicine, there is a lack of focus on AI-driven solutions for doctor selection in this field.

This research gap is particularly concerning due to the personalized nature of Ayurvedic treatment, which requires precise matching of patients with doctors based on individual symptoms. Currently, patients lack technological assistance in navigating the complex Ayurvedic medicine landscape, hindering their ability to make informed choices.

Furthermore, the incorporation of patient feedback and ratings into the selection process of Ayurvedic doctors is crucial for informed decision-making. Existing mobile applications in this area often underutilize user feedback and ratings to suggest suitable doctors. AI-driven systems could significantly enhance this process by incorporating patient perspectives, ultimately improving healthcare access quality.

Feature	Existing Mobile Applications	Proposed System
AI-powered doctor selection based on symptoms	No	Yes

AI-powered doctor selection based on proximity	No	Yes
AI-powered doctor selection based on ratings/feedback	No	Yes
Integration of patient feedback and ratings	Limited	Yes
Ability to book appointments online	Yes	Yes
Information on doctor qualifications and experience	Yes	Yes
Information on doctor availability and fees	Yes	Yes
Specific focus on Ayurvedic medicine	Limited	Yes

Table 4: Research gap summary 13

Table provides a succinct comparison between existing mobile applications in this domain and the proposed system, highlighting the key features that the latter would introduce to address the identified research gap. Notably, the proposed system not only integrates AI for doctor selection based on symptoms, proximity, and user feedback but also offers the convenience of online appointment booking, comprehensive information on doctor qualifications, experience, availability, and fees, all with a specific focus on Ayurvedic medicine.

Social networks are increasingly used for health-related information and support. However, there is a need for natural language processing solutions to extract insights from the vast amounts of textual data. Currently, there is a lack of comprehensive solutions tailored to health-based social networks, and developing models for this context is challenging. Duplicate content and sentiment analysis are also essential but underexplored tasks. Privacy and ethical concerns must be addressed in designing the integrated solution.

2 RESEARCH OBJECTIVES

2.1 Main Objectives

One of the primary objectives of this study is to develop a user-friendly mobile application dedicated to promoting a healthier lifestyle through Ayurveda, specifically focusing on Ayurvedic treatments and lifestyle guidelines for common symptoms such as arthritis, blood sugar management, hair loss, infertility, obesity, paranasal sinusitis, and minor injuries. This application will feature a conversational AI chatbot, utilizing a robust knowledge base derived from trusted Ayurvedic sources to offer personalized recommendations and herbal remedies based on users' specific symptoms. It will also provide valuable dietary and lifestyle guidance while emphasizing the importance of consulting healthcare professionals for comprehensive health management. Regular updates, user engagement features, and strict adherence to legal and ethical considerations will be integral to ensuring the application's effectiveness and user satisfaction.

Another main objective of this research is to develop a creative solution to promote a healthier lifestyle by utilizing the principles of Ayurveda by addressing the common symptoms for some diseases through identifying appropriate Ayurvedic medical herbs. Additionally, a component based on a geometry library will be implemented to map out the locations of these identified herbs which will make easier for the patient to find herbs.

To achieve this objective, the proposed solution will consist of an image processing-based component to identify the necessary herbal plants for treating these diseases. Furthermore, the solution will incorporate continual learning/transfer learning techniques to enhance the accuracy of the system, and an Auto Machine Learning component to facilitate the addition of new plants to the system. Ultimately, the objective of this research is to provide individuals with a more accessible and comprehensive approach to achieving a healthier lifestyle through Ayurveda.

Another central aim of this comprehensive study is to conceive, develop, and launch a cutting-edge mobile application tailored to the unique needs of Ayurvedic healthcare seekers. At its core, this transformative application seeks to revolutionize the way patients engage with Ayurvedic medicine by addressing several fundamental objectives:

1. **Symptom-Based Doctor Identification:** The primary objective is to empower users to input their specific symptoms into the application. Leveraging advanced AI algorithms, the application will then meticulously analyze these symptoms to identify the most appropriate Ayurvedic doctor for personalized treatment. This fundamental feature ensures that each patient's healthcare journey is finely tuned to their individual needs and health concerns.
2. **User Ratings and Feedback:** In the pursuit of excellence, the application will seamlessly integrate user ratings and feedback mechanisms. This dynamic functionality not only empowers patients to share their valuable insights but also plays a pivotal role in the selection of the best Ayurvedic doctors. Users will have the agency to contribute their experiences, fostering transparency and accountability within the Ayurvedic healthcare ecosystem.
3. **Proximity-Based Doctor Listings:** Recognizing the importance of convenience, the application will utilize geolocation services to present users with a curated list of nearby Ayurvedic doctors. This ensures that patients have easy access to practitioners within their vicinity, facilitating timely and hassle-free healthcare services.
4. **Interactive Communication:** To nurture trust and facilitate seamless interactions, users will have the ability to engage in private chats with their chosen Ayurvedic doctors. This feature fosters open lines of communication, enabling patients to seek guidance, clarification, and personalized advice directly from their healthcare provider.
5. **Appointment Booking:** Convenience lies at the heart of this application. Users will be able to effortlessly schedule appointments with their selected Ayurvedic doctors through the platform. This streamlines patient-provider interaction, ensuring that healthcare services are readily accessible and well-organized.

To facilitate the realization of these transformative objectives, it is imperative that both patients and Ayurvedic practitioners engage with the application as registered users. This ensures the security, privacy, and seamless functioning of the platform, creating a trusted environment where healthcare decisions are made with confidence.

Another main objective is to implement a social network that will be limited to sharing health-related content. Users can both ask questions and post articles and can respond to them as well. S. Detect near duplications of the content against existing public content in the social network to

ensure that users do not repeatedly post highly similar or identical content, reducing redundancy and maintaining content quality.

1. Implement an enhanced semantic search functionality to provide users with more accurate and comprehensive results for health-related topics while understanding the context and intent behind user queries. Moreover, search results will be improved by considering user preferences and historical interactions.
2. Implement an information extraction pipeline to automatically extract and analyze health-related trends from the social network's content and save in a knowledge base.

2.2 Specific Objectives

For the chat bot, we have followed these specific objectives,

The development process for the proposed software application focused on non-communicable diseases (NCDs) and Ayurvedic treatments involves several critical stages:

1. **Research and Data Gathering:** Thorough research is essential to gather comprehensive information on selected non-communicable diseases, including their causes, symptoms, and conventional medical treatments. Additionally, gathering Ayurvedic knowledge related to these diseases is crucial, as it forms the foundation for the AI model.
2. **Knowledge Base Creation:** The collected information is structured and organized to create a robust knowledge base. This database contains data on symptoms, common health problems associated with each NCD, and the corresponding Ayurvedic treatments and medicinal herbs for these conditions.
3. **Data Pre-processing and Cleaning:** Data pre-processing involves cleaning and formatting the information to make it suitable for AI modeling. This step ensures data consistency and accuracy by handling missing values, outliers, and inconsistencies.
4. **AI Model Training:** With a prepared database, the AI model is trained using machine learning techniques. The model learns to identify patterns and relationships between symptoms, common health problems, and Ayurvedic treatments based on the provided data.
5. **Evaluation and Validation:** To assess the AI model's performance, it is tested on a validation dataset. Evaluation metrics are used to measure its accuracy, precision, recall, and other

relevant criteria. This step helps identify areas for improvement.

6. **Fine-Tuning:** Based on the evaluation results, the model is fine-tuned to enhance its accuracy and effectiveness. This iterative process may involve adjusting hyperparameters, optimizing algorithms, and expanding the dataset.
7. **Software Application Development:** The AI model is integrated into a user-friendly software application or website. This application will serve as a platform for users to interact with the AI system and receive personalized Ayurvedic recommendations for managing NCDs.
8. **Verification and Expert Consultation:** The software's accuracy and validity are verified by consulting Ayurvedic experts and practitioners. Their feedback and expertise ensure that the recommendations align with traditional Ayurvedic principles and are safe and effective.
9. **Conversational AI Chatbot:** A key feature of the software application is the conversational AI chatbot. This chatbot allows users to engage in text-based conversations, providing a user-friendly and accessible interface for receiving personalized solutions and advice for NCDs through Ayurveda.

By following these steps, the software application aims to bridge the gap between modern healthcare and Ayurveda, offering users a reliable resource for managing non-communicable diseases using holistic and traditional Ayurvedic approaches. The integration of AI technology and expert verification ensures the accuracy and effectiveness of the recommendations provided to users, ultimately contributing to improved health and well-being.

For the herb identification collection of images of different ayurvedic herbs and labeling them.

1. Pre-processing of the collected images, such as resizing and normalization
2. Splitting the data into training, validation, and testing sets.
3. Training a machine learning model using pre-processed image data.
4. Evaluating the model on the validation set to identify areas of improvement.
5. Collecting and processing large amounts of image data from various geographical areas to map the distribution of herbal plants.
6. Fine-tuning the model based on the evaluation results.
7. Testing the final model on the testing set to measure its accuracy and robustness.
8. Implementing the model in a software application for practical use.

As novelty improvement,

Using continual learning/transfer learning to improve accuracy. (Also, can incorporate Auto Machine Learning to make it easier to add new plants.)

To achieve the overarching goal of enhancing accessibility and personalization in Ayurvedic healthcare, the following specific objectives have been meticulously outlined:

1. **Develop an Intuitive and User-Friendly Mobile Application Interface:** Create an aesthetically pleasing and user-centric interface that simplifies the process of inputting symptoms and offers a seamless navigation experience for users.
2. **Implement a Robust Machine Learning Model:**
 - Deploy a sophisticated machine learning model capable of analyzing a patient's symptoms comprehensively.
 - Utilize this model to intelligently recommend the most suitable Ayurvedic doctors based on factors such as specialization, ratings, and proximity to the patient's location.
3. **Integrate a Feedback and Rating System:**
 - Incorporate a feedback and rating system within the application, empowering patients to rate their interactions and experiences with Ayurvedic doctors.
 - Leverage this user-generated feedback to enhance the recommendation system continually.
4. **Develop a Private Chat Feature:**
 - Create a secure and user-friendly private chat feature that facilitates direct communication between patients and Ayurvedic doctors.
 - Enable patients to seek guidance, ask questions, and schedule appointments effortlessly.
5. **Establish a Robust Registration System:**

- Implement a comprehensive registration system that allows both patients and Ayurvedic doctors to create accounts within the application.
 - Enable users to manage their profiles effectively, providing essential information for personalized healthcare.
6. Implement a Secure Payment System: Develop a secure and efficient payment system within the application, enabling patients to make payments for appointments and consultations seamlessly.
 7. Create Appointment Booking Functionality:
 - Build a feature that allows patients to check the real-time availability of Ayurvedic doctors.
 - Enable patients to book appointments based on their preferred schedule, enhancing convenience.
 8. Establish a Centralized Medical History Database:
 - Develop a comprehensive database that securely stores the medical histories of patients.
 - Provide Ayurvedic doctors with seamless access to this database during consultations, ensuring a holistic understanding of the patient's health.
 9. Ensure Regulatory Compliance:
 - Ensure that the application complies with all relevant regulatory requirements, including data privacy laws and healthcare standards.
 - Prioritize the security and confidentiality of patient data.
 10. Continuously Enhance User Experience:
 - Conduct rigorous user testing and actively solicit feedback from users to identify areas of improvement.

- Commit to ongoing refinement and optimization of the application's functionality and user interface to ensure a superlative user experience. These specific objectives collectively contribute to the development of a comprehensive and user centric Ayurvedic healthcare application, aligning with the overarching mission of improving healthcare accessibility and personalization.

For the social network implementation,

1. Detect near duplications of the content against existing public content in the social network to ensure that users do not repeatedly post highly similar or identical content, reducing redundancy and maintaining content quality.
2. Implement an enhanced semantic search functionality to provide users with more accurate and comprehensive results for health-related topics while understanding the context and intent behind user queries. Moreover, search results will be improved by considering user preferences and historical interactions.
3. Implement an information extraction pipeline to automatically extract and analyze health-related trends from the social network's content and save in a knowledge base.

3 METHODOLOGY

3.1 Requirement gathering and Analysis.

- Collecting information from Gampaha Wickramarachchi Ayurvedic University

To collect information on ayurveda and diseases we met **Dr. Janaki Wickramarachchi**, who is the Dean of Chikisthsaka Faculty at **Gampaha Wickramarachchi Ayurvedic University** and had conducted some online meetings conducted with her, with participation of our group members. She agreed to provide us the necessary information related to the ayurveda and the research gap which is having when connecting with modern technologies such as Artificial Intelligence and Machine Learning. She highlighted several main diseases which are fine for the research. She gave us legal approval for the continuation of the research and gave advice about the things we need to focus on in the future while continuing the project.

- Data gathering

Firstly, we read a dozen published research for initial understanding and got some basic idea by reading and browsing through few articles. Our supervisors had few meetings with us to discuss the initial methods for data gathering and the external supervisor connected us with a few ayurvedic specialties and pointed out the diseases and the data we are needed continue with. In future the other necessary data and images will be collected from the University of Gampaha as necessities.

- Conducting a survey

To get an idea about the knowledge of people about ayurvedic treatments and diseases and their knowledge about the connection between AI/ML with it, we have conducted a survey was conducted with both closed and open-ended questions by distributing a questionnaire.

3.2 Feasibility studies

Economic feasibility is a critical aspect of any project's success, as it determines whether or not the project is financially viable. The economic feasibility report analyzes the development costs and benefits of the project, and if a proper economic feasibility plan is not in place, the project is likely to fail. Therefore, it is crucial that the proposed system is both cost-effective and efficient in order to ensure its success.

- Scheduled feasibility

Scheduled feasibility is another essential factor to consider when undertaking a project. A schedule feasibility assessment examines the timelines for the planned project, and any delays or missed deadlines can have a significant impact on the project's success. Therefore, it is vital that the proposed system completes each task within the allotted time period as specified to ensure that the project stays on schedule.

- Technical feasibility

Technical feasibility planning is also crucial in the development of any system. It involves evaluating the required skills and expertise necessary for mobile and web application development,

as well as the ability to understand software architectures and communicate effectively with stakeholders to obtain the necessary information. Without proper technical feasibility planning, it is unlikely that the proposed system will be successfully developed and implemented. Therefore, it is essential to have the necessary technical skills and communication abilities to move forward with the system's development.

3.3 System Diagrams

3.3.1 Overall system diagram

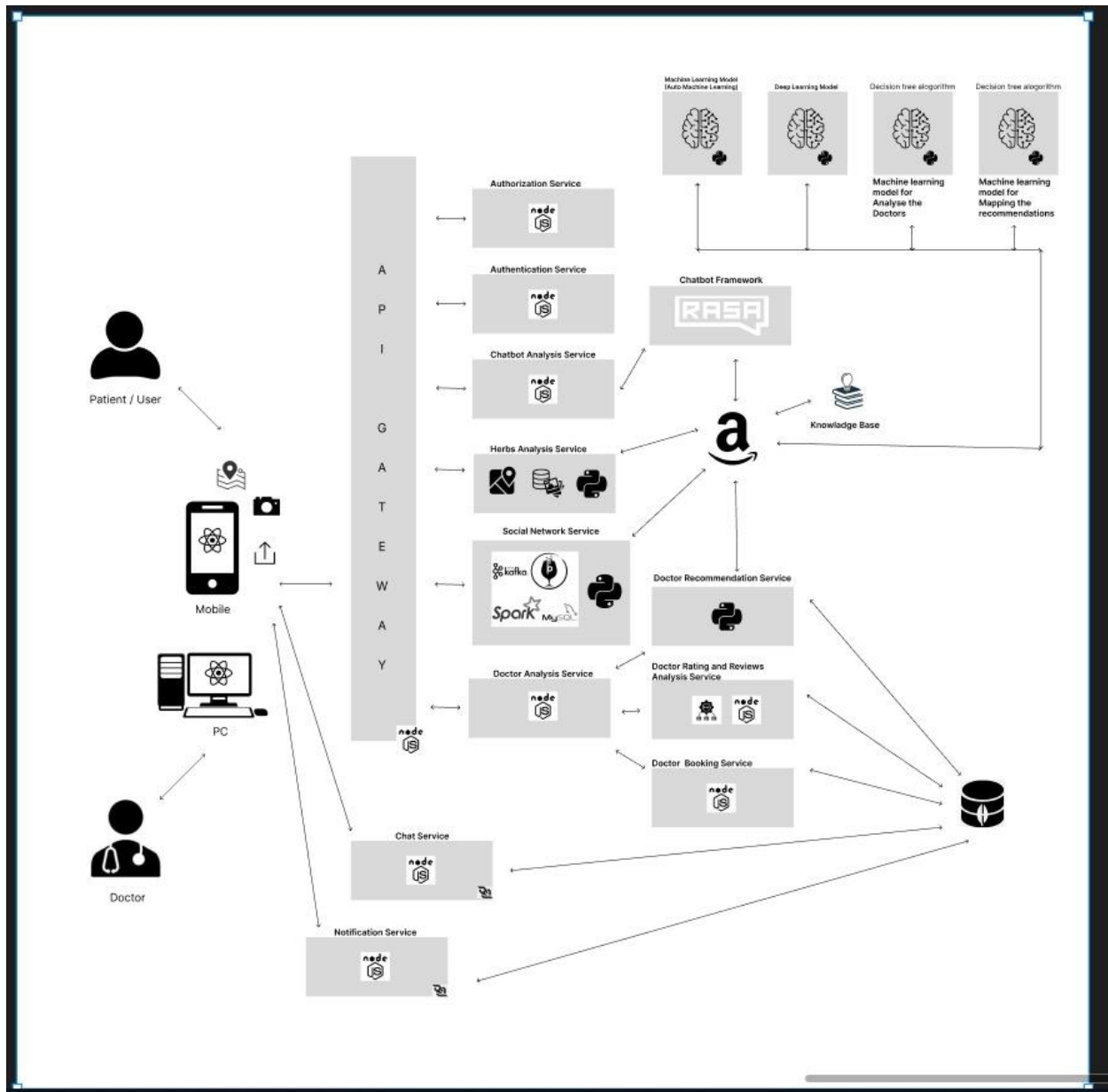


Figure 14: Overall System Architecture Diagram

3.3.2 Individual component diagrams

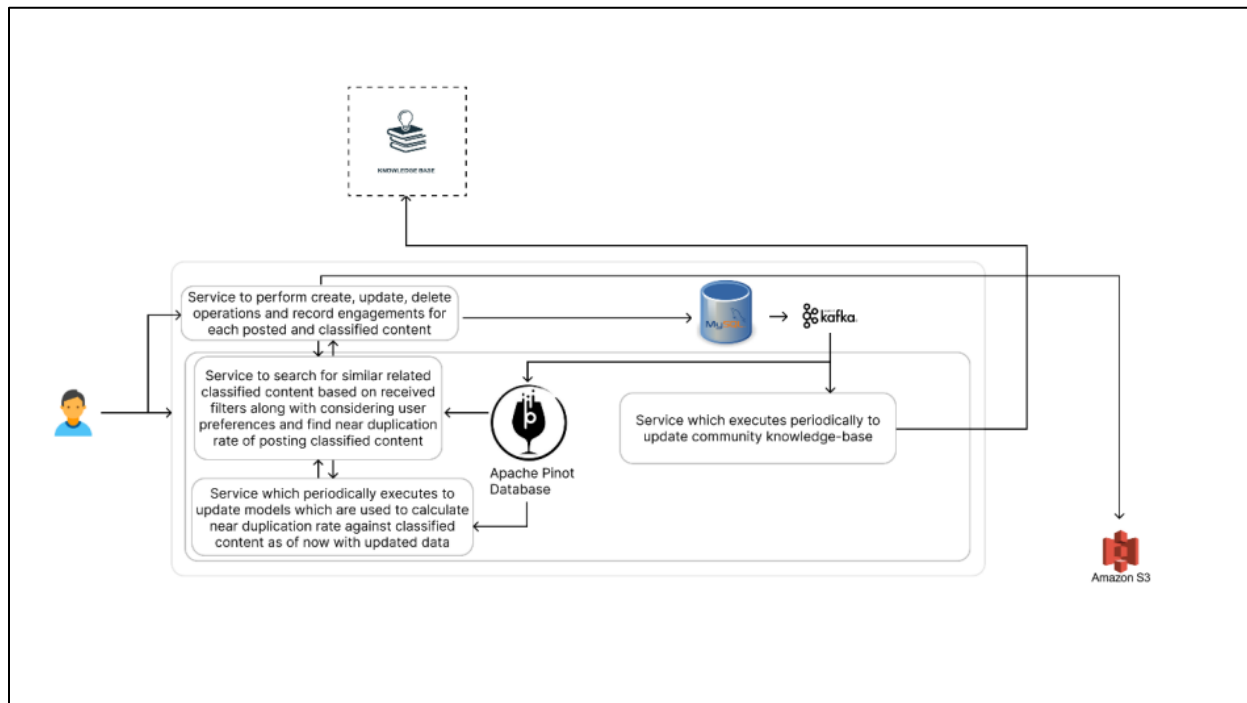


Figure 15: Individual component diagram

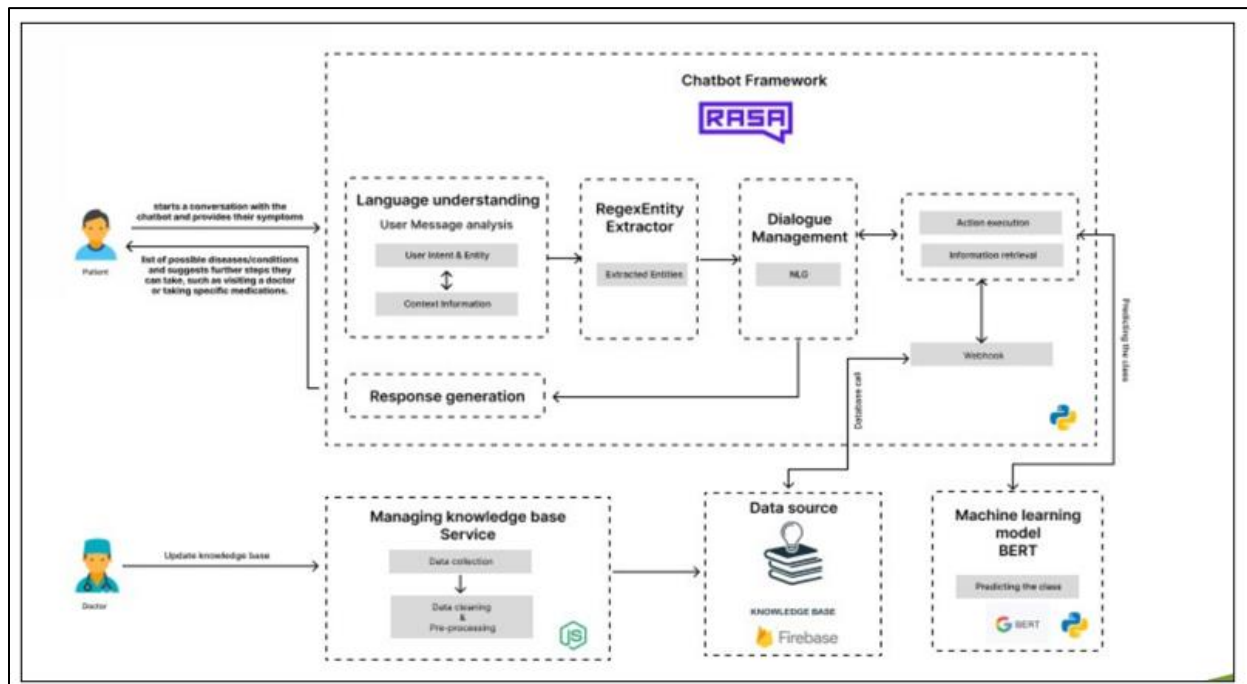


Figure 17: Individual component diagram

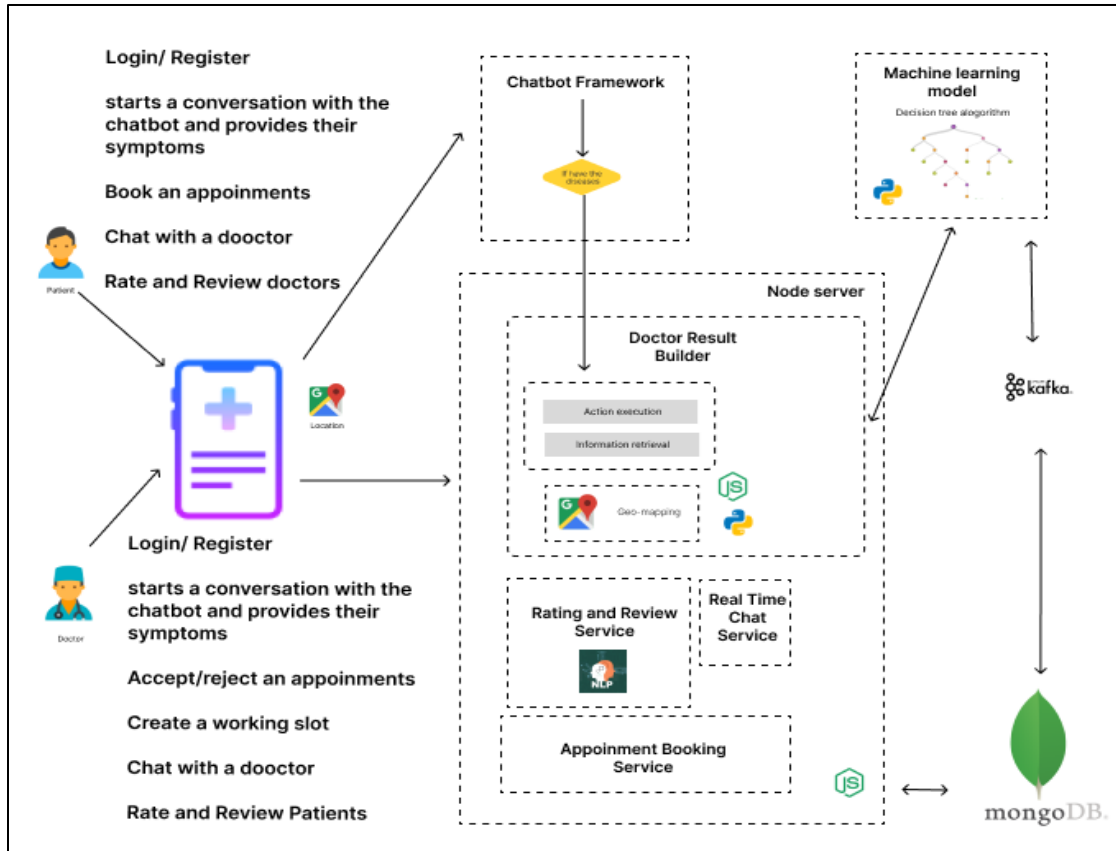


Figure 18: Individual component diagram

3.4 Understanding the Key Pillars of the Research Domain

3.4.1 Machine learning

Machine learning (ML) represents a paradigm shift in problem-solving, offering a solution to complex challenges where the manual development of algorithms by human programmers is economically unfeasible. Instead, ML empowers machines to autonomously uncover algorithms without explicit human guidance. Recent advancements in the field have seen generative artificial neural networks surpassing previous methods across various domains. ML techniques find applications in diverse fields, including large language models, computer vision, speech recognition, email filtering, agriculture, and medicine, addressing tasks that would otherwise be prohibitively costly to tackle through conventional algorithmic development.

The mathematical foundation of ML is closely tied to mathematical optimization techniques, providing the essential framework for its methodologies. In parallel, data mining explores the realm of exploratory data analysis through unsupervised learning, contributing to the broader landscape of data-driven insights.

Within the realm of business problem-solving, ML assumes the guise of predictive analytics. While ML encompasses a range of approaches, not all of which rely on statistical principles, computational statistics plays a significant role in shaping the methodologies applied in the field.

The roots of machine learning trace back to the pursuit of artificial intelligence (AI). In the early stages of AI's academic development, researchers sought ways to enable machines to learn from data. They explored a variety of symbolic methods and what were initially known as "neural networks," primarily focusing on perceptron's and models that later evolved into generalized linear models in statistics. The use of probabilistic reasoning, especially in fields like automated medical diagnosis, was also prevalent.

Over time, a divergence emerged between the logical, knowledge-based AI approach and machine learning. Challenges related to data acquisition and representation plagued probabilistic systems. By the 1980s, expert systems had taken center stage in AI, sidelining statistics. Symbolic and knowledge-based learning persisted within AI, resulting in inductive logic programming, while statistical research in pattern recognition and information retrieval branched away from AI. Neural networks research was simultaneously abandoned in both AI and computer science domains. However, researchers like Hopfield, Rumelhart, and Hinton continued their work, coining the term "connectionism" and achieving significant breakthroughs, such as the reinvention of backpropagation.

The 1990s marked a renaissance for machine learning, as it emerged as an independent discipline. The field shifted its focus from the grand goal of achieving artificial intelligence to solving practical, tangible problems. It departed from the symbolic AI methods inherited from the past, embracing techniques borrowed from statistics, fuzzy logic, and probability theory. Machine learning models can be broadly categorized into three paradigms based on the nature of available signals or feedback.

Supervised learning involves the computer learning a general rule that maps inputs to outputs, guided by examples provided by a "teacher." Unsupervised learning, on the other hand, unfolds when no explicit labels are given to the learning algorithm, prompting it to autonomously uncover underlying patterns in the input data. Reinforcement learning introduces dynamic interaction with an environment, where the program strives to achieve specific objectives, akin to navigating and

maximizing rewards. Each learning paradigm has its strengths and limitations, with no single algorithm capable of solving all conceivable problems.

3.4.2 Natural Language Processing (NLP)

The domain of computer science recognized as "natural language processing" (NLP) falls specifically under the purview of "artificial intelligence" (AI). It is primarily concerned with bestowing computers with the capability to comprehend written and spoken language in a manner reminiscent of human understanding.

NLP amalgamates statistical, machine learning, and deep learning models with the field of computational linguistics, which involves constructing rule-based representations of human language. Through the utilization of these technologies, computers now possess the ability to process human language in the form of textual or auditory data and achieve a profound "comprehension" of the content, encompassing the intentions and emotional nuances conveyed by the speaker or writer.

NLP underpins various computer applications, encompassing those that facilitate the translation of text between languages, respond to spoken directives, and swiftly distill extensive textual information, even in real-time scenarios. Chances are, you have interacted with NLP in everyday consumer products, including voice-activated GPS devices, digital personal assistants, speech-to-text transcription programs, automated customer service chatbots, and a spectrum of other user-friendly utilities. Furthermore, the deployment of NLP in corporate solutions is expanding, serving as an instrumental means of optimizing business operations, amplifying workforce productivity, and streamlining pivotal business processes.

Creating software that can reliably discern the intended meaning from text or voice data is an exceptionally daunting task due to the inherent complexities of human language. Human communication is fraught with ambiguity, including homonyms, homophones, sarcasm, idiomatic expressions, metaphors, deviations from grammatical conventions, and shifts in sentence structures. These intricacies, which take humans years to master, must be imparted to natural language-driven applications from the outset to ensure their utility. To facilitate the computer's comprehension of the text and speech data it processes, several NLP tasks are employed. Here are some of these tasks.

1. **Speech Recognition (Speech-to-Text):** This process involves accurately transcribing voice data into text and is commonly known as speech recognition or speech-to-text. Any program that responds to voice commands or inquiries relies on speech recognition. The challenges arise from the diverse ways individuals speak—quickly, with words slurred together, varying emphases and intonations, diverse dialects, and sometimes non-standard grammar usage—all of which make speech recognition particularly challenging.
2. **Part of Speech Tagging (Grammatical Tagging):** Part of speech tagging entails determining the grammatical category of a word based on its usage and context. For example, in the sentences "I can make a paper plane" and "What make of car do you own?," the word "make" is categorized as a verb and a noun, respectively.
3. **Word Sense Disambiguation:** This task involves selecting the correct meaning of a word from among its potential meanings by conducting semantic analysis to discern which meaning aligns with the context at hand. Word sense disambiguation is crucial for distinguishing between different senses of a word, such as "make" in "make the grade" (achieve) and "make a bet" (place).
4. **Named Entity Recognition (NER):** NER identifies and classifies words or phrases as specific entities, such as recognizing "Kentucky" as a location or "Fred" as a person's name.
5. **Co-reference Resolution:** Co-reference resolution revolves around determining whether and when two words refer to the same entity. It often involves resolving pronoun references (e.g., determining that "she" refers to "Mary"), but can also encompass identifying metaphors or idioms used in the text (e.g., understanding that "bear" refers to a large, hairy person rather than the animal).
6. **Sentiment Analysis:** Sentiment analysis seeks to uncover nuanced elements in text, including emotions, attitudes, sarcasm, confusion, and mistrust.
7. **Natural Language Generation:** Natural language generation is the process of transforming structured data into human language. It is essentially the opposite of voice recognition or speech-to-text, where machines generate human-like text based on data inputs.

3.4.3 Convolutional Neural Network (CNN)

CNNs are subsets of machine learning. It is one of the many varieties of artificial neural networks used for CNNs, or Convolutional Neural Networks, constitute a specific network architecture within the realm of deep learning algorithms, primarily tailored for tasks related to image recognition and the processing of pixel data. While deep learning encompasses various types of neural networks, CNNs emerge as the preferred architecture for the precise identification and recognition of objects. This inherent capability positions them as the optimal choice for tasks in the domain of computer vision (CV), particularly in applications where robust object recognition plays a pivotal role, such as in autonomous vehicles and facial recognition systems.

Deep learning algorithms extensively rely on artificial neural networks (ANNs) to model complex patterns and relationships. Among the diverse ANN types, recurrent neural networks (RNNs) stand out, specifically designed for processing sequential or time series data. RNNs find their niche in natural language processing (NLP), machine translation, speech recognition, and image captioning applications, where sequential information is paramount.

What sets CNNs apart is their versatility in extracting critical insights from both time series and image data. This versatility significantly benefits tasks involving images, including but not limited to image recognition, object classification, and pattern identification. CNNs leverage fundamental principles of linear algebra, such as matrix multiplication, to discern intricate patterns within images. Moreover, they extend their classification prowess to audio and signal data domains.

The structural arrangement of a CNN closely mimics the connectivity pattern observed in the human brain. Like the brain, CNNs comprise billions of neurons strategically organized. Remarkably, the neurons within a CNN bear resemblance to the frontal lobe of the human brain, responsible for processing visual stimuli. This specific configuration ensures comprehensive coverage of the entire visual field, eliminating the issue of fragmented image processing that plagues conventional neural networks, requiring low-resolution image inputs in disjointed segments. Consequently, CNNs exhibit superior performance when presented with image inputs, as well as when dealing with speech or audio signal inputs, surpassing the capabilities of older network architectures.

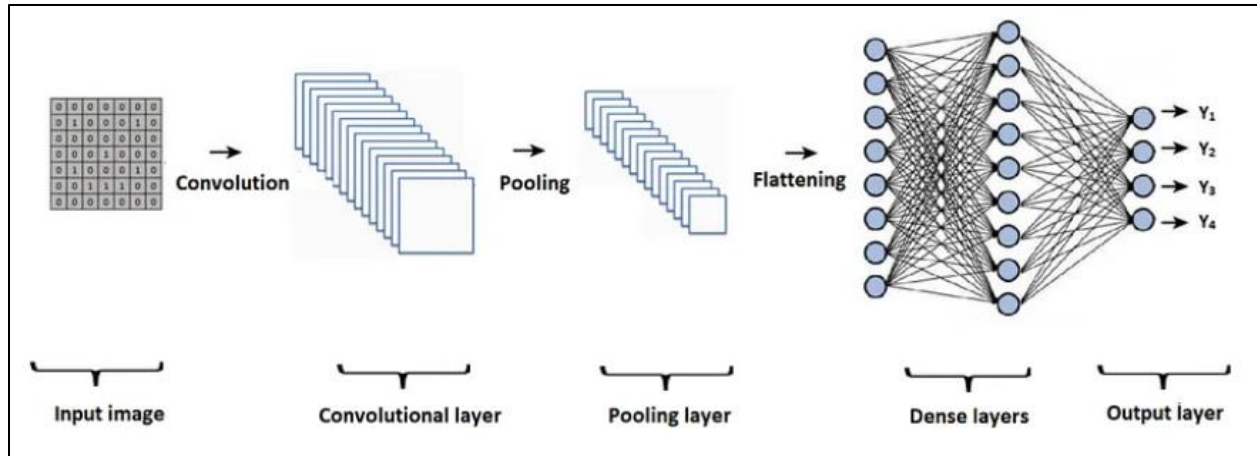


Figure 16: CNN Architecture

A deep learning Convolutional Neural Network (CNN) is structured with three distinct layers: the convolutional layer, the pooling layer, and the fully connected (FC) layer. The convolutional layer is the initial layer, while the FC layer serves as the final layer in the network.

As the CNN progresses from the convolutional layer towards the FC layer, its complexity steadily increases. This incremental complexity empowers the CNN to progressively recognize larger segments and intricate characteristics within an image, ultimately leading to the complete identification of objects.

The convolutional layer serves as the cornerstone of the CNN, responsible for a significant portion of computations. This layer often includes multiple convolutional operations, and it all begins with the initial convolutional layer. During the convolution process, a kernel or filter within this layer traverses the image's receptive fields to assess the presence of specific features. Over multiple iterations, this kernel systematically explores the entire image, calculating the dot product between input pixels and the filter after each pass. The cumulative result of these computations is known as a feature map or convoluted feature, effectively converting the image into numerical values. This numerical representation enables the CNN to comprehend the image and discern relevant patterns.

The pooling layer, similar to the convolutional layer, employs a kernel or filter to traverse the input image. However, in contrast to the convolutional layer, the pooling layer serves to reduce the

number of input parameters and introduces some degree of information loss. On the positive side, this layer simplifies the network's structure and enhances its efficiency.

The CNN's FC layer is where the pivotal task of image classification takes place, leveraging the extracted features from preceding layers. In this context, "fully connected" implies that all inputs or nodes from one layer establish connections with every activation unit or node in the subsequent layer.

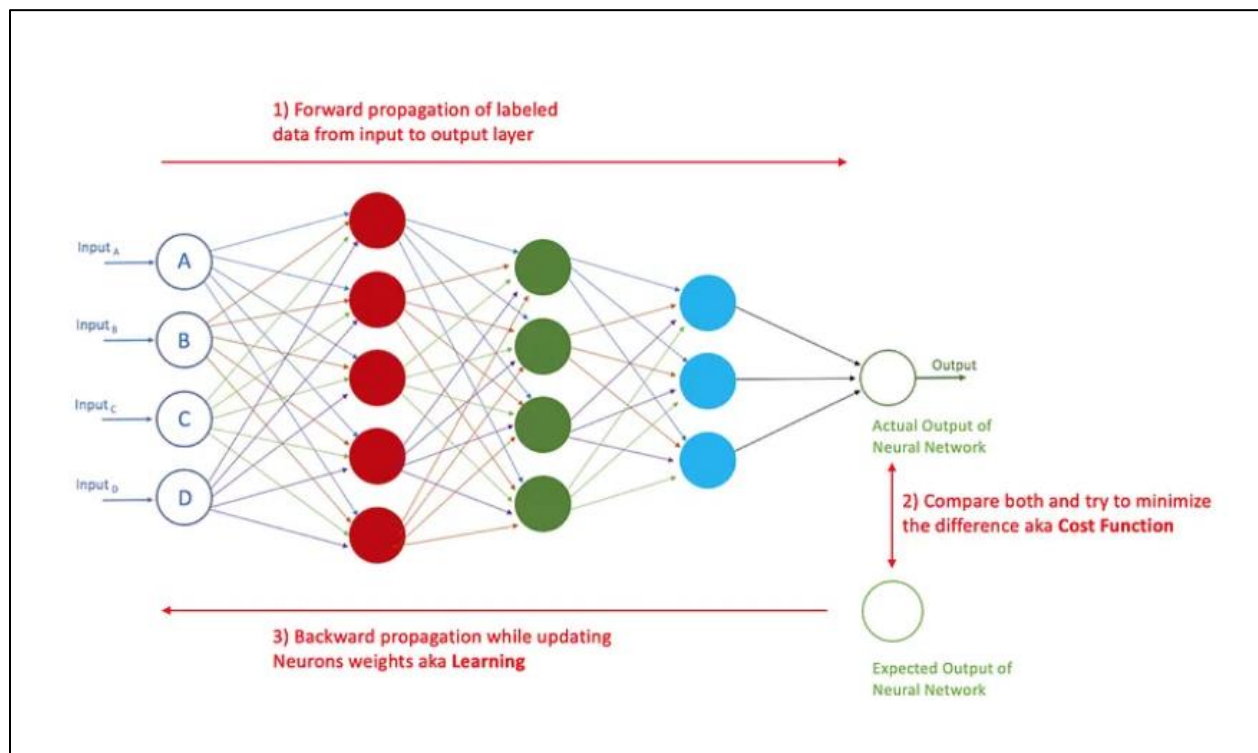


Figure 17: Process of Model Training

CNN avoids integrating all of its layers as such integration would lead to an unnecessarily dense network, potentially resulting in increased attrition, reduced output quality, and elevated computational demands.

A CNN can encompass multiple layers, and each of these layers is dedicated to learning and detecting distinct image features. In this process, a filter or kernel is applied to the input image, gradually refining and enhancing the output as it progresses through successive layers. In the earlier layers, these filters typically begin by detecting fundamental features.

With each subsequent layer, the complexity of the filters intensifies to thoroughly scrutinize and pinpoint the distinguishing characteristics that define the input object uniquely. Consequently, the output of each convolved image, representing a partially recognized image after each layer's operation, serves as the input for the subsequent layer in the hierarchy. Ultimately, in the final layer, referred to as the Fully Connected (FC) layer, the CNN achieves the identification of the image or object it represents.

The convolution process entails the application of these filters to the input image, with each filter selectively activating specific image features while transmitting its output to the subsequent layer's filter. Each layer specializes in recognizing distinct features, leading to the repetitive execution of these operations across numerous layers. The cumulative information from all the image data that has traversed through the multiple layers enables the CNN to accomplish the holistic identification of the entire object.

3.4.4 Data augmentation

The precision of a deep learning model depends on the quantity, quality, and relevance of training data. However, a scarcity of data is one of the most common obstacles in machine learning applications. In many situations, the collection of such information can be expensive and time-consuming.

Data augmentation is the process of producing additional data points from extant data in order to augment the size of an existing dataset.

Creating an extended dataset can be accomplished by making minor modifications to existing data or by employing deep learning models to generate additional data points. Companies can use data augmentation to reduce their need to acquire and prepare training data. This allows them to construct more accurate machine learning models more quickly.

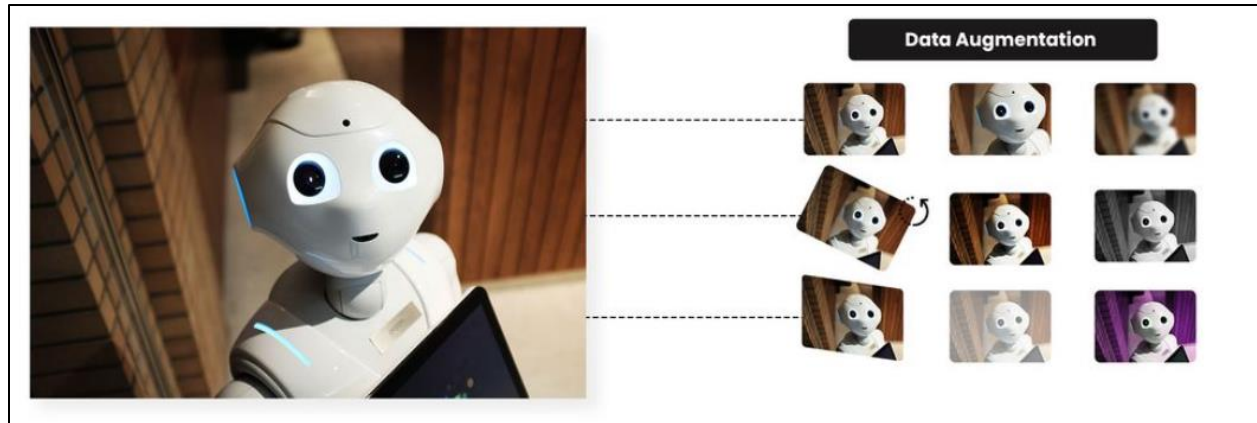


Figure 18:Data Augmentation

3.5 Specific Methodology

The methodology to achieve these objectives involves:

- Collect a dataset of plant images: You will need a dataset of labeled images of plants for training your machine learning model.
- Preprocess the data: You will need to preprocess the data by resizing the images, normalizing the pixel values, and splitting the dataset into training and validation sets.
- Feature extraction: You will need to extract features from the images that can be used to train your machine learning model. There are several feature extraction techniques available, such as Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), and Convolutional Neural Networks (CNNs).
- Train the machine learning model: Once you have extracted the features, you can train your machine learning model. There are several algorithms you can use for classification, including convolutional Neural Networks.
- Test and Evaluate: After training the model, you will need to test it using a separate dataset and evaluate its performance. You can use metrics such as accuracy, precision, recall, and F1 score to evaluate the performance of the model.
- Deploy the system: Finally, you can deploy the system and make it accessible to users.

This element has been incorporated into the system to enhance user interactions related to herbal remedies. Its primary aim is to assist users in comprehending and identifying Ayurvedic herbs.

This feature enables users to obtain detailed information about herbs recommended by the chatbot, empowering them to explore these herbs independently. When users receive herbal suggestions from the chatbot, they can access an interface that provides extensive details about these recommended herbs. Users can also manually search for herbs based on the information provided. Additionally, a unique functionality allows users to capture images of herbs through their device's camera if they encounter a specific herb that is challenging to identify. This capability is implemented through a combination of backend and frontend functionalities.

On the backend side, the Fast API framework is employed to create an interface that accepts image uploads and initiates the herb prediction process. Uploaded images are processed, and a trained model is employed to predict the herb that best matches the user's input. This model utilizes a pre-trained VGG19 architecture that has been fine-tuned using a dataset comprising images of herbs. The integration with Firebase ensures that comprehensive herb details, including images and additional information, are stored and retrieved seamlessly. Users can access this comprehensive herb information stored in the database, aiding them in identifying the encountered herbs.

On the frontend aspect, users interact with the image upload interface. Users have the option to upload images of herbs they've found, and the system provides predictions based on these images. If a user cannot find an exact match for the identified herb, they can use the camera feature to capture images for analysis. Additionally, users can access a dedicated image upload area, offering a straightforward and intuitive method for submitting herb images. The backend model processes these uploaded images, generating predictions that help users identify the herbs they've discovered. This interactive feature bridges the gap between the chatbot's recommendations and user-driven exploration, fostering an empowering and educational experience.

- **Model Training Phase:**
 - **Dataset Creation:** A diverse and comprehensive dataset of Ayurvedic herb images is curated, encompassing various herb species and perspectives.
 - **Data Preprocessing:** Images are standardized by resizing them to a standard dimension (e.g., 224x224 pixels) while maintaining the aspect ratio. Pixel values are normalized to a specific range (usually [0, 1]).
 - **Dataset Split:** The dataset is divided into training and validation sets for effective model

evaluation.

- **Model Architecture:** A pre-trained neural network architecture, such as VGG19, is selected as the base model due to its robust feature extraction capabilities.
- **Transfer Learning:** The pre-trained model's convolutional base serves as a feature extractor, and new dense layers are added on top for classification.
- **Fine-tuning and Training:** The entire model is fine-tuned on the Ayurvedic herb dataset. Convolutional layer weights are frozen, while dense layer weights are updated during training.
- **Validation and Hyperparameter Tuning:** Model performance is evaluated on the validation dataset, and hyperparameters like learning rate, batch size, and epochs are fine-tuned for optimal results.
- **Herb Prediction (Inference) Phase:**
 - **Image Preprocessing:** When a user uploads an herb image, it undergoes preprocessing steps to align with the model's requirements.
 - **Model Inference:** The pre-trained and fine-tuned herb identification model predicts the herb species from the preprocessed image, providing a probability distribution over herb classes.
 - **Class Selection:** The herb class with the highest predicted probability is chosen as the final prediction, providing the identified herb's name.
 - **Herb Details Retrieval:** Using the predicted herb class label, detailed information about the identified herb is retrieved from the Firebase database, including textual details and related images.
 - **Presentation to Users:** The identified herb and its associated data are displayed through the frontend interface, allowing users to explore the details and properties of the recommended herb based on the provided information.

This research component follows a structured approach encompassing the development of the backend herb identification model, the creation of a user-friendly image upload interface, seamless integration with Firebase for herb information storage, and the provision of comprehensive herb details to users. Evaluation of this component involves assessing the accuracy of herb predictions and measuring user satisfaction with the feature.

3.6 Commercialization aspects of the system

This study offers a creative answer in the form of a potential filling of one of the research gaps that were discovered by our research. Everyone who has a stake in the matter, including patients, ayurveda practitioners working in the ayurvedic industry, and any other parties that are pertinent, will be informed about the application known as "AYUR MANA." The Gampaha Wickramarachchi Ayurvedic University holds events on a regular basis for university students and practitioners. Because the institution has its own medicinal supply systems across the nation, these events also feature the presentation of products in an effort to boost sales in the local community.

In addition to that, promotion of AYUR MANA will take place on the official website of the Gampaha Wickramarachchi Ayurvedic University. There is already an application available in the Google Play store that customers who are interested in downloading can use on their portable technology.

Patients, ayurvedic practitioners, and members of the general public who have an interest in ayurveda are the primary target stakeholders that this system intends to accommodate.

3.7 Consideration of the aspect of the system

During the process of coding each individual component, coding standards were adhered to.

3.7.1 Social aspects

The smartphone application known as 'AYUR MANA' can be utilized by anyone with varying degrees of familiarity and expertise with technology. The early diagnosis and management of diseases and pests will help save enormous economic losses, which in turn will contribute to the reduction of global poverty and the improvement of people's access to nutritious food.

The process of early infestation control has been made more difficult since there are not enough tools that are based on information and communications technology to assist in real-time collaboration between producers and researchers. Therefore, by the use of this cutting-edge technology, we can provide a way for early detection and quick communication while also ensuring the education of the farmers, which will ultimately assist in the survival of the stakeholders for sustainable ayurvedic solutions.

3.7.2 Security aspects

The smartphone application known as "AYUR MANA" provides users with the ability to safeguard the confidentiality of their private information, detection details, and user authentication. Concerns regarding the level of security were taken into account right from the beginning of the page. For the purpose of ensuring the user's authenticity, JWT tokens were utilized. While communicating over the network, several ways for encrypting data were utilized in order to circumvent the security checks. Requests for user rights were made in order to improve the system's integrity and security.

3.7.3 Ethical aspects

The design of the system takes into consideration several ethical factors. There is no upper age limit for users of this platform.

Children are able to successfully use this program as well. The culture is respected, and the application has not yielded any negative feedback in that regard.

4 IMPLEMENTATION AND TESTING

4.1 Implementation

The research project offers a comprehensive suite of software solutions encompassing both web and mobile applications, designed to address critical aspects of image identification, classification, and progression level assessment in herb plants related to ayurveda. These applications are collectively known as "AYUR MANA."

- **AYUR MANA Mobile Application:**

Purpose: The mobile application, "AYUR MANA," serves as a versatile tool for the identification, classification, and evaluation of disease progression in coconut palms.

Functionality:

- **Disease Identification:** AYUR MANA allows users to identify plant leaves affecting various diseases, providing valuable insights into potential threats to the humans.
- **Classification:** The application enables the classification of identified leaves, aiding in the categorization and differentiation.

- **Progression Level Calculation:** Users can assess the severity and progression level of diseases, facilitating timely intervention and management strategies.

Implementation Technology: The front end of the mobile application is implemented using React Native, a cross-platform framework known for its efficiency and ability to deliver consistent user experiences across different mobile platforms.

- **AYUR MANA Web Application:**

Purpose: The web application is designed to cater to the specific needs of researchers at the Gampaha Wikramarachchi Ayurvedic University, offering continuous monitoring capabilities for disease-infected patients across the country.

Functionality:

- **Monitoring:** practitioners can monitor the status of plant herbs in real-time, facilitating data-driven decision-making.
- **Data Analysis:** The web application supports the analysis of leaves and patterns, aiding researchers in gaining valuable insights.

Implementation Technology:

1. **Frontend Development:** The frontend of the web application is implemented using React.js, a popular JavaScript library known for its flexibility and performance in creating dynamic user interfaces.
2. **Backend Development:** Node.js is employed for the backend development of the web application. Node.js is well-suited for handling real-time data updates and providing a robust server-side environment.

In summary, AYUR MANA is a comprehensive software solution comprising a mobile application for on-the-ground plant leaf identification and assessment and a web application for remote monitoring and data analysis. The technologies chosen for implementation, such as React Native, React.js, and Node.js, ensure efficiency, cross-platform compatibility, and real-time data handling,

making AYUR MANA a powerful tool for researchers and stakeholders involved in the management and preservation of Ayurvedic Medicine palm lands.

4.1.1 Model implementation

4.1.1.1 Chatbot

```
▷ # Load your dataset
data = pd.read_csv("output.csv")

# Preprocessing and data formatting
class CustomDataset(Dataset):
    def __init__(self, data, tokenizer, max_length):
        self.data = data
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        text = self.data["Syntomes "].iloc[idx]
        label = int(self.data["Disease"].iloc[idx])

        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            truncation=True,
            max_length=self.max_length,
            padding="max_length",
            return_tensors="pt",
            return_attention_mask=True,
            return_token_type_ids=False,
        )

        return {
            "input_ids": encoding["input_ids"].flatten(),
            "attention_mask": encoding["attention_mask"].flatten(),
            "labels": torch.tensor(label, dtype=torch.long),
        }
```

[2]

Figure 19: Code Snippet of BERT Model

This code snippet begins by loading a dataset from a CSV file named "output.csv" using the Pandas library. This dataset likely contains valuable information about symptoms and their associated disease labels. Following this, the code defines a custom dataset class called **CustomDataset** to prepare the data for training a BERT-based model. Inside this class, there are essential methods such as `__init__`, `__len__`, and `__getitem__`. The `__init__` method initializes the dataset with the dataset itself, a BERT tokenizer, and a maximum sequence length. The `__len__` method specifies

the length of the dataset based on the number of rows in the DataFrame. The most crucial part, the `__getitem__` method, retrieves an item from the dataset at a given index. It extracts both the text (representing symptoms) and the corresponding disease label from the DataFrame. The BERT tokenizer is then applied to encode the text, incorporating special tokens, managing truncation, and padding. Finally, the method returns a dictionary containing the tokenized input text, an attention mask indicating padded tokens, and the disease label as a torch tensor. Overall, this code prepares and formats the data for training a BERT-based sequence classification model, facilitating the utilization of the dataset in machine learning tasks related to disease prediction based on symptom descriptions.

```
# Training loop
model.train()
for epoch in range(epochs):
    total_loss = 0
    for batch in train_loader:

        input_ids = batch["input_ids"].to(device)
        attention_mask = batch["attention_mask"].to(device)
        labels = batch["labels"].to(device)

        optimizer.zero_grad()
        outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        total_loss += loss.item()

        loss.backward()
        optimizer.step()

    avg_loss = total_loss / len(train_loader)
    print(f"Epoch {epoch + 1}/{epochs}, Loss: {avg_loss:.4f}")
```

13]

```
.. Epoch 1/5, Loss: 0.0298
    Epoch 2/5, Loss: 0.0008
    Epoch 3/5, Loss: 0.0009
    Epoch 4/5, Loss: 0.0008
    Epoch 5/5, Loss: 0.0000
```

Figure 20: Code Snippet of training loop

In here, the training loop for the BERT-based model is defined. The model is set to training mode using `model.train()`, which enables gradient computation and weight updates during training. The

loop iterates over the specified number of epochs, where each epoch represents a complete pass through the training dataset. Within each epoch, a total loss variable is initialized to track the cumulative loss for that epoch.

For each batch of data in the training loader (provided by **train_loader**), the input tensors (**input_ids**, **attention_mask**, and **labels**) are moved to the selected computing device (e.g., GPU) for efficient computation. The optimizer's gradients are zeroed using **optimizer.zero_grad()** to avoid gradient accumulation between mini-batches. Then, the model is called with the input tensors, and it computes both the logits and the loss with **outputs = model(...)**. The loss is extracted from the outputs.

The computed loss is accumulated in **total_loss**, and backpropagation (**loss.backward()**) is performed to compute gradients with respect to the model's parameters. The optimizer (**optimizer.step()**) then updates the model's weights using the computed gradients. This process is repeated for all mini-batches in the training dataset.

At the end of each epoch, the average loss for that epoch is calculated by dividing **total_loss** by the number of mini-batches (**len(train_loader)**), providing insight into the model's performance during training. This average loss is printed, allowing you to monitor the training progress across epochs. Overall, this code snippet encapsulates the core training logic, where the model learns to make predictions and adjust its parameters to minimize the defined loss function.


```

def pred(input_text):
    # Tokenize the input text
    input_encoding = tokenizer.encode_plus(
        input_text,
        add_special_tokens=True,
        truncation=True,
        max_length=20,
        padding="max_length",
        return_tensors="pt",
        return_attention_mask=True,
        return_token_type_ids=False,
    )

    # Move input tensors to the same device as the model (if available)
    input_ids = input_encoding["input_ids"].to(model.device)
    attention_mask = input_encoding["attention_mask"].to(model.device)

    # Make the prediction
    model.eval()
    with torch.no_grad():
        outputs = model(input_ids, attention_mask=attention_mask)
        logits = outputs.logits

    # Move logits to the CPU (if needed)
    logits = logits.cpu()

    # Get the predicted class
    predicted_class = torch.argmax(logits, dim=1).item()
    print("Predicted Class:", predicted_class)

```

Figure 21: Code Snippet of training loop

The **pred** function is designed to make predictions using the fine-tuned BERT-based model. It begins by tokenizing the input text using the BERT tokenizer. The **encode_plus** method takes care of this tokenization, ensuring that the input text is appropriately converted into a format suitable for the model. Key parameters, such as adding special tokens, truncating or padding sequences to a maximum length of 20 tokens, and returning tensors and attention masks, are specified.

The input tensors, **input_ids** and **attention_mask**, are then moved to the same computing device as the model, whether it's a CPU or GPU. This step ensures that the input data is processed on the correct hardware.

The model is set to evaluation mode using **model.eval()**, which disables gradient computation and speeds up inference. With the input tensors, predictions are made using the model. Specifically, the logits (raw output scores) are obtained from the model's forward pass.

To ensure compatibility, the logits are moved to the CPU using **logits.cpu()**. Finally, the predicted class is determined by selecting the class index with the highest probability using **torch.argmax(logits, dim=1).item()**. This index represents the model's prediction for the input text. The predicted class is printed to the console for reference.

4.1.1.2 Herb identification

```
def add_data():
    rootdir= 'new_data'
    sub_class = os.listdir(rootdir)

    for classes in sub_class:
        try:
            os.mkdir("./Dataset/train/"+classes)
            os.mkdir("./Dataset/test/"+classes)
            os.mkdir("./Dataset/resize train/"+classes)
            os.mkdir("./Dataset/resize test/"+classes)
            print("Data Added")
        except:
            print("Already in the system")

    for classes in sub_class:
        print(classes)
        if is_folder_empty("Dataset/train/"+classes):
            print(f"The folder '{classes}' is empty.")
            path_to_class = rootdir + '/' + classes
            allimgs = os.listdir(path_to_class)
            np.random.shuffle(allimgs)
            test_ratio = 0.25
            train_FileNames, test_FileNames = np.split(np.array(allimgs),[int(len(allimgs)* (1 - test_ratio))])

            train_FileNames = [path_to_class+'/'+ name for name in train_FileNames.tolist()]
            test_FileNames = [path_to_class+'/'+ name for name in test_FileNames.tolist()]

            for name in train_FileNames:
                shutil.copy( name , './Dataset/train/' + classes)

            for name in test_FileNames:
                shutil.copy(name, './Dataset/test/' + classes)
        else:
            print(f"The folder '{classes}' is not empty.")
```

Figure 22: Code Snippet to splitting data into training and test data

The `add_data` function appears to perform the following tasks related to organizing and splitting data into training and testing datasets for multiple classes:

1. Creating directories

- It first lists the subdirectories (classes) within the "new_data" directory using `os.listdir(rootdir)`.
- For each class, it attempts to create several subdirectories inside the "Dataset" directory:
- `"./Dataset/train/" + class_name`
- `"./Dataset/test/" + class_name`
- `"./Dataset/resize train/" + class_name`
- `"./Dataset/resize test/" + class_name`
- If the subdirectories already exist, it prints "Already in the system." If they don't exist, it

prints "Data Added."

2. Data splitting

- For each class, it checks if the "Dataset/train/class_name" directory is empty using the ``is_folder_empty`` function. If it's empty, it proceeds to split the data into training and testing sets.
- It shuffles the list of image filenames within the class directory and defines a `test_ratio` (e.g., 0.25, indicating a 25% test set).
- It then splits the list of image filenames into two lists: ``train_FileNames`` and ``test_FileNames``, according to the specified test ratio.
- For each filename in ``train_FileNames``, it copies the image file to `"/Dataset/train/class_name."`
- For each filename in ``test_FileNames``, it copies the image file to `"/Dataset/test/class_name."`

3. Handling non-empty directories

- If the "Dataset/train/class_name" directory is not empty (indicating that data splitting may have already occurred), it prints "The folder 'class_name' is not empty."
- This code is essentially organizing and splitting data into training and testing datasets for each class, assuming that each class's data is stored in separate subdirectories under the "new_data" directory. It also handles cases where the data splitting has already been performed for some classes.
- The ``data_pre_processing`` function appears to perform the following tasks related to image data preprocessing:

```

def data_pre_processing():
    IMG_SIZE = (227,227)
    src_train_path = './Dataset/train/'
    dst_train_path = './Dataset/resize train/'
    src_test_path = './Dataset/test/'
    dst_test_path = './Dataset/resize test/'

    train_dir = os.listdir(src_train_path)
    test_dir = os.listdir(src_test_path)

    for img_list in train_dir:
        x = src_train_path+img_list

        img_folders = os.listdir(x)

        for img_name in img_folders:
            img_dst_train_path = dst_train_path + '/' + img_list + '/' + img_name
            img_src_train_path = src_train_path + '/' + img_list + '/' + img_name
            img = Image.open(img_src_train_path)
            img = img.resize(IMG_SIZE,Image.LANCZOS)
            img.save(img_dst_train_path)

    for img_list in test_dir:
        x = src_test_path+img_list
        img_folders = os.listdir(x)

        for img_name in img_folders:
            img_dst_test_path = dst_test_path + '/' + img_list + '/' + img_name
            img_src_test_path = src_test_path + '/' + img_list + '/' + img_name
            img = Image.open(img_src_test_path)
            img = img.resize(IMG_SIZE,Image.LANCZOS)
            img.save(img_dst_test_path)

```

Figure 23: Code Snippet to Data Processing

4. Image Size Configuration-

- It sets the desired image size to (227, 227) pixels as `IMG_SIZE`. This indicates that all images will be resized to this specified dimension.

5. Source and Destination Paths Configuration

- It defines source and destination paths for both training and testing datasets:
- `src_train_path`: The source directory containing the original training images.
- `dst_train_path`: The destination directory where resized training images will be saved.
- `src_test_path`: The source directory containing the original testing images.
- `dst_test_path`: The destination directory where resized testing images will be saved.

6. Listing Directories

- It lists the directories (classes) present in the training and testing datasets:
- `train_dir`: List of classes in the training dataset.

- `test_dir`: List of classes in the testing dataset.

7. Resizing Images

- It iterates through each class in the training and testing datasets (`train_dir` and `test_dir`).
- For each class, it lists the images (`img_folders`) present in the respective source directory.
- It then iterates through each image in the class.
- For each image, it constructs the source path (`img_src_train_path` for training and `img_src_test_path` for testing) and destination path (`img_dst_train_path` for training and `img_dst_test_path` for testing) based on the class and image name.
- It opens the original image using the Python Imaging Library (PIL), resizes it to the specified `IMG_SIZE` using Lanczos interpolation, and saves the resized image to the destination path.
- In summary, the `data_pre_processing` function resizes all the images in the training and testing datasets to a specified size (227x227 pixels) using Lanczos interpolation and saves the resized images in separate directories within the "resize train" and "resize test" directories, respectively. This preprocessing step is common in image data preparation for

```
def model_building():
    IMAGE_SIZE = [224, 224]
    vgg19 = VGG19(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
    for layer in vgg19.layers:
        layer.trainable = True
    folders = glob('Dataset/train/*')
    x = Flatten()(vgg19.output)
    prediction = Dense(len(folders), activation='softmax')(x)
    model = Model(inputs=vgg19.input, outputs=prediction)
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    print("Image Data Generator")
    train_datagen = ImageDataGenerator(rescale = 1./255,
                                       shear_range = 0.2,
                                       zoom_range = 0.2,
                                       horizontal_flip = True)

    test_datagen = ImageDataGenerator(rescale = 1./255)
    training_set = train_datagen.flow_from_directory('Dataset/train',
                                                    target_size = (224, 224),
                                                    batch_size = 50,
                                                    class_mode = 'categorical')

    test_set = test_datagen.flow_from_directory('Dataset/test',
                                                target_size = (224, 224),
                                                batch_size = 50,
                                                class_mode = 'categorical')

    print("model training ")
    r = model.fit(
        training_set,
        validation_data=test_set,
        epochs=30,
        steps_per_epoch=len(training_set),
        validation_steps=len(test_set))
```

Figure 24: Code Snippet to model building

machine learning tasks.

- The `model_building` function appears to define and train a convolutional neural network (CNN) model for image classification using the VGG19 architecture and Keras. Here's an explanation of what this function is doing:
- **Image Size Configuration:**
 - 1) `IMAGE_SIZE = [224, 224]`: It sets the desired input image size for the model to 224x224 pixels.
 - 2) Loading the VGG19 Model:
 - 3) `vgg19 = VGG19(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)`: This line loads the VGG19 model pretrained on the ImageNet dataset. It specifies the input shape, loads pretrained weights, and excludes the fully connected layers (`include_top=False`).
- **Setting VGG19 Layers as Trainable:**
 - 1) `for layer in vgg19.layers: layer.trainable = True`: It sets all layers in the VGG19 model to be trainable.
- **Gathering Class Labels:**
 - 1) `folders = glob('Dataset/train/*')`: It gathers the class labels from the training dataset by listing subdirectories under "Dataset/train."
- **Building the Custom Model:**
 - 1) `x = Flatten()(vgg19.output)`: It flattens the output of the VGG19 base model.
 - 2) `prediction = Dense(len(folders), activation='softmax')(x)`: It adds a fully connected layer with softmax activation for classification.
 - 3) `model = Model(inputs=vgg19.input, outputs=prediction)`: It constructs the custom model by defining the input and output layers.
 - 4) `model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])`: It compiles the model, specifying the loss function, optimizer, and evaluation metric.
- **ImageDataGenerator for Data Augmentation:**
 - 1) It sets up data augmentation for the training dataset and rescaling for the

testing dataset using ImageDataGenerator.

- **Loading Training and Testing Data:**

- 1) `training_set = train_datagen.flow_from_directory('Dataset/train', ...)`: It loads the training data using ImageDataGenerator and specifies the target size, batch size, and class mode.
- 2) `test_set = test_datagen.flow_from_directory('Dataset/test', ...)`: It loads the testing data in a similar manner.

- **Model Training:**

- 1) `model.fit(...)`: It trains the model on the training data and validates it on the testing data. The training includes multiple epochs, steps per epoch, and validation steps.

- **Saving the Model:**

- 1) `model.save('model_3.h5')`: It saves the trained model to a file named "model_3.h5."
- 2) In summary, this function builds, compiles, and trains a CNN model for image classification using transfer learning with the VGG19 architecture. The training data is augmented for better generalization, and the trained model is saved to a file for future use.


```
add_data()
```

```
Already in the system
Already in the system
Already in the system
Already in the system
Already in the system
Already in the system
Already in the system
Alpinia Galanga (Rasna)
The folder 'Alpinia Galanga (Rasna)' is not empty.
Amaranthus Viridis (Arive-Dantu)
The folder 'Amaranthus Viridis (Arive-Dantu)' is not empty.
Artocarpus Heterophyllus (Jackfruit)
The folder 'Artocarpus Heterophyllus (Jackfruit)' is not empty.
Azadirachta Indica (Neem)
The folder 'Azadirachta Indica (Neem)' is not empty.
Basella Alba (Basale)
The folder 'Basella Alba (Basale)' is not empty.
Brassica Juncea (Indian Mustard)
The folder 'Brassica Juncea (Indian Mustard)' is not empty.
test
The folder 'test' is not empty.
```

```
data_pre_processing()
```

```
model_building()
```

Figure 25:ode Snippet to image classification model using deep learning

These functions appear to be part of a larger pipeline for building and training an image classification model using deep learning techniques. By executing these functions in sequence, you are organizing your data, preparing it for model training, and then training the model itself.

4.1.1.3 Doctor recommendation

```
from sklearn.tree import DecisionTreeClassifier
import os
import joblib

# Define the path to save and backup the model
MODEL_DIR = 'models'
MODEL_FILE = 'doctor_recommendation_model.pkl'
model_path = os.path.join(MODEL_DIR, MODEL_FILE)
CSV_FOLDER = 'csv_data'

# Check if the CSV folder exists, if not, create it
if not os.path.exists(CSV_FOLDER):
    os.makedirs(CSV_FOLDER)

# Define the file name for the CSV file
CSV_FILE_NAME = 'encoded_data.csv'

# Define the full path for the CSV file
CSV_FILE_PATH = os.path.join(CSV_FOLDER, CSV_FILE_NAME)

# Check if the model directory exists, if not, create it
if not os.path.exists(MODEL_DIR):
    os.makedirs(MODEL_DIR)

clf = DecisionTreeClassifier()
clf.fit(X_encoded, y)

# Save the trained model to the model directory
model_path = os.path.join(MODEL_DIR, MODEL_FILE)
joblib.dump(clf, model_path)
```

Figure 26: Model Implementation code snippet

This code snippet is responsible for training a Decision Tree Classifier model and saving it for future use. Initially, it defines key file paths and directories, such as where to save the trained model (MODEL_DIR and MODEL_FILE) and where to store CSV data (CSV_FOLDER and CSV_FILE_NAME). It also checks if these directories exist and creates them if not. Next, a Decision Tree Classifier is instantiated, and it's trained using previously prepared and encoded data stored in X_encoded and y. The training step is where the machine learning model learns from the data. After training, the code saves the trained model to a file specified by model_path. This model persistence step is crucial for later use, as it allows the trained model Figure 13: Model Implementation code snippet 29 to be loaded and applied to make predictions without the need for retraining. Overall, this code ensures that the trained machine learning model is organized and stored in a designated directory for easy access and future use.

```

const { LanguageServiceClient } = require("@google-cloud/language");

// Create a client instance
const client = new LanguageServiceClient();

const analyzeSentiment = async ({ comment }) => {
  try {
    // Define the document content
    const document = {
      content: comment,
      type: "PLAIN_TEXT",
    };

    // Analyze sentiment
    const [result] = await client.analyzeSentiment({ document });

    // Get sentiment score
    const sentimentScore = result.documentSentiment.score;

    // Determine sentiment label
    let sentimentLabel = "";
    if (sentimentScore >= 0.25) {
      sentimentLabel = "Positive";
    } else if (sentimentScore <= -0.25) {
      sentimentLabel = "Negative";
    } else {
      sentimentLabel = "Neutral";
    }

    // console.log('Text: ${comment}');
    // console.log('Sentiment Score: ${sentimentScore}');
    // console.log('Sentiment Label: ${sentimentLabel}');
    return sentimentScore;
  } catch (error) {
    console.error("Error:", error);
    throw new Error(error);
  }
};

module.exports = analyzeSentiment

```

Figure 27::Sentiment Score Analysis code snippet

This JavaScript code defines a function called `analyzeSentiment` that leverages Google Cloud's Natural Language API for sentiment analysis. It begins by importing the necessary module, `LanguageServiceClient`, and creating a client instance named `client` to interact with the API. The core function, `analyzeSentiment`, accepts an object parameter with a `comment` property representing the text to be analyzed. Within the function, the provided text is structured into a document, specifying its content and type as plain text. The sentiment of this document is then analyzed using the `client.analyzeSentiment()` method, and the result is stored in the `result` variable. The sentiment score is extracted from the result, representing the overall sentiment of the text. Depending on this score, the function determines a sentiment label: "Positive" for scores above or equal to 0.25, "Negative" for scores below or equal to -0.25, and "Neutral" for values in between. The code also includes error handling using a `try...catch` block to log and handle any potential errors that may arise during the sentiment analysis process. Finally, the `analyzeSentiment` function is exported as a module, making it available for use in other parts of a Node.js application. This code simplifies the task of sentiment analysis, allowing developers to easily

Score Analysis code snippet 30 assess the sentiment of text comments using the Google Cloud Natural Language API by invoking this function.

```
const R = 6371; // Radius of the Earth in kilometers.

function calculateDistance(lat1, lon1, lat2, lon2) {
  const dlat = (lat2 - lat1) * (Math.PI / 180);
  const dlon = (lon2 - lon1) * (Math.PI / 180);

  const a =
    Math.sin(dlat / 2) * Math.sin(dlat / 2) +
    Math.cos(lat1 * (Math.PI / 180)) *
      Math.cos(lat2 * (Math.PI / 180)) *
      Math.sin(dlon / 2) *
      Math.sin(dlon / 2);

  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

  const distance = R * c;
  return distance;
}

function sortCoordinatesByDistance(referenceLat, referenceLon, coordinates) {
  // Calculate distance for each coordinate and store it in an array of objects
  const distances = coordinates.map((coord) => {
    const { latitude, longitude } = coord;
    const distance = calculateDistance(
      referenceLat,
      referenceLon,
      latitude,
      longitude
    );
    return { coord, distance };
  });

  // Sort the distances array by distance
  @distances.sort((a, b) => a.distance - b.distance);

  // Extract the sorted coordinates
  const sortedCoordinates = distances.map((item) => item.coord);

  return sortedCoordinates;
}
```

Figure 28:Distance Sorter Algorithm

This JavaScript code defines two functions for calculating distances between geographical coordinates on the Earth's surface and sorting a list of coordinates based on their proximity to a reference point. It utilizes the Haversine formula for distance calculations, which is a wellknown formula for calculating the great-circle distance between two points on the Earth's surface, given their latitude and longitude coordinates. The calculateDistance function takes four parameters: lat1 and lon1 representing the latitude and longitude of the first point, and lat2 and lon2 representing the latitude and longitude of the second point. It begins by converting the differences in latitude and longitude from degrees to radians. Then, it uses the Haversine formula to calculate the great-circle distance between the two points.

```

// Custom sorting function
function customSort(a, b) {
    // First, compare by rate in descending order
    if (a.rate > b.rate) return -1;
    if (a.rate < b.rate) return 1;

    // If rates are equal, compare by sentiment score in descending order
    if (a.positiveSentimentScore > b.positiveSentimentScore) return -1;
    if (a.positiveSentimentScore < b.positiveSentimentScore) return 1;

    // If both rate and sentiment score are equal, no change in order
    return 0;
}

// Sort the data using the custom sorting function
doctorDataList.sort(customSort);

```

Figure 29: Rating and Reviews Sorter Algorithm code snippet

This JavaScript code defines a custom sorting function named `customSort` designed for sorting a list of objects based on multiple criteria. Here's a breakdown of how the code works:

Custom Sorting Function The `customSort` function takes two objects, `a` and `b`, as input parameters. These objects are typically elements from a list that you want to sort. The primary sorting criterion is the `'rate'` property of these objects, which represents a numerical rating. The function first compares these `'rate'` values in descending order. If `a` has a higher `'rate'` than `b`, it returns `-1` (indicating that `a` should come before `b` in the sorted list). If `a` has a lower `'rate'`, it returns `1` (indicating that `a` should come after `b`). If the `'rate'` values are equal, the function proceeds to the next criterion.

Secondary Sorting Criterion: If the `'rate'` values of `a` and `b` are equal, the function compares their `'positiveSentimentScore'` properties. This represents a sentiment score, and again, it sorts in descending order. Higher sentiment scores come first, and lower scores come later in the sorted list.

Equality Handling: If both the `'rate'` and `'positiveSentimentScore'` are equal between `a` and `b`, the function returns `0`, signifying that there is no change in the order of these objects. This ensures that elements with identical `'rate'` and `'positiveSentimentScore'` values maintain their relative order.

Sorting Data: After defining the custom sorting function, the code applies it to a list of objects called `doctorDataList` using the `sort` method. By calling `doctorDataList.sort(customSort)`, the list is sorted based on the defined criteria. In summary, this code provides a flexible way to sort a list of objects by multiple criteria, primarily by `'rate'` and, if necessary, by `'positiveSentimentScore'`. It ensures that the list is arranged in descending order of `'rate'`, and in cases where `'rate'` values are

equal, it further refines the order based on 'positiveSentimentScore'. This can be useful in various scenarios, such as ranking items or professionals based on both their ratings and sentiment scores.

4.1.1.4 Social network

Enhanced Semantic Search Functionality The enhanced search functionality of Sentence-Transformers semantic search is utilized to make social network searches more convenient. There are two approaches to semantic search.

Symmetric search: In symmetric semantic search tasks, both the query and the entries within your dataset are typically of similar length and contain a comparable amount of content. To illustrate this, consider searching for similar questions. For instance, your query might be "How to learn Python online?" with the goal of finding an entry such as "How to learn Python on the web?" In these scenarios, it's often feasible to interchange the query and the entries within your dataset to achieve the desired results.

Asymmetric search: Asymmetric semantic search tasks, the query is usually short, such as a question or a set of keywords, and the objective is to locate a longer paragraph or document that provides an answer or detailed information related to the query. For example, a query like "What is Python?" would aim to retrieve a paragraph like "Python is an interpreted, high-level, and general-purpose programming language. Python's design philosophy..." In asymmetric tasks, reversing the positions of the query and the entries in your dataset typically does not yield meaningful results.

It used asymmetric search implementation to deliver better search results for the users. The semantic search functionality implemented for the proposed social network consists of several steps.

The initial step in the data collection process involves obtaining unprocessed text data from social media platforms, which may include various forms of user-generated content such as questions, articles, and responses. This data is then queried from the Apache Pinot database.

To begin the process of encoding the user query, a pre-trained "all-MiniLML6-v2" sentence-transformer model is initialized. This model is known for its fast-processing speed, high accuracy and performance.

Using the loaded Sentence Transformer model, the received user query is encoded to obtain its embedding. Similarly, the collected data from the database is encoded using the same model to obtain embeddings for each content.

The next step is to search for closer content from the collected data for the received user query, based on cosine similarity. The results are returned in a paginated format, providing an efficient and effective way to access relevant content. Information Extraction Pipeline to Extract Trending Health Information The implementation of the information extraction pipeline involves the utilization of several natural language processing techniques and previously conducted research to efficiently detect current health trends within the proposed social network. If we consider a sentence such as "Sugar has a negative impact on diabetes, but it is beneficial for low blood sugar," a series of steps can be proposed to extract the required information from a health-based social network.

Data preprocessing: Each queried content or response will be considered separately in this step. o First, each queried text will be processed to remove punctuation using regular expressions. o Next, the processed texts in the previous step will undergo a process to get the singular form of each word. It uses used NLTK WordNetLemmatizer module for this functionality

```
def get_singular_form(word):  
    # Get the singular form of a word using WordNet lemmatizer  
    lemmatizer = WordNetLemmatizer()  
    singular_form = lemmatizer.lemmatize(  
        word, pos="n"  
    ) # 'n' specifies that it is a noun  
    return singular_form
```

Figure 30:Process Singular Form of a Word

Next, the processed texts in the previous step will undergo a coreference resolution process. It is used spaCy “neuralcoref” coreference resolution module for this functionality.

```

def coref_resolution(text):
    doc = nlp(text)
    # fetches tokens with whitespaces from space document
    tok_list = list(token.text_with_ws for token in doc)
    for cluster in doc._coref_clusters:
        # get tokens from representative cluster name
        cluster_main_words = set(cluster.main.text.split(" "))
        for coref in cluster:
            if (
                coref != cluster.main
            ): # if coreference element is not the representative element of that cluster
                if (
                    coref.text != cluster.main.text
                    and not(
                        set(coref.text.split(" ")).intersection(cluster_main_words)
                    )
                ):
                    # if coreference element text and representative element text are not equal and none of the coreference element words
                    # are in representative element. This was done to handle nested coreference scenarios
                    tok_list[coref.start] = {
                        cluster_main.text + doc[coref.end - 1].whitespace,
                    }
                    for i in range(coref.start + 1, coref.end):
                        tok_list[i] = ""
    return "".join(tok_list)

```

Figure 31:Process Co-Reference Resolution

Tokenization: This step in processing a sentence is to tokenize it into individual words or tokens. This process, also known as tokenization, can be achieved using Python programming language's built-in methods, as outlined in the proposed solution. o Tokenize the sentence: ["Sugar", "is", "negatively", "affecting", "diabetes", "while", "it", "is", "good", "for", "low-blood-sugar", "."]

Part-of-Speech (POS) Tagging: Apply part-of-speech tagging to identify the grammatical roles of each token. This helps distinguish between entities and sentiment-related modifiers.

Dependency Parsing: Use dependency parsing to understand the grammatical relationships between words. This step helps establish connections between entities and sentiment modifiers.

Named Entity Recognition: Identify and extract entities mentioned in the sentence. Use named entity recognition (NER) to recognize entities

Sentiment Analysis: Analyze the sentiment of each clause. Identify words or modifiers that convey sentiment.

Relationship Extraction: Based on the dependency parsing and the understanding of the sentence, extract relationships between entities and sentiment

Overall Analysis: Combine the extracted information to create a structured representation of the sentence.

4.2 Testing

4.2.1 Test plan and test strategy

The testing strategy specifies the aspects to be tested, and the functions to be tested are selected based on the importance of the functions and the risks they pose to users. The test cases were then written under the available use cases, they were manually handled, and the results were recorded. Test planning is crucial for creating a baseline plan with tasks and accomplishments to track the project's progress. It also demonstrates the test's scope.

1. Employed test Strategy:
2. Define the testable items.
3. Choose the functions based on user risk and relevance.
4. design test cases in accordance with the use case description
5. execute
6. record results
7. find bugs.
8. fix bugs

and repeat the test case until the desired results are obtained.

5 RESULTS AND DISCUSSIONS

5.1 Results

5.1.1 For the chatbot

The disease classification model has yielded highly promising results, marking a significant advancement in the capabilities of the Ayurvedic recommendation chatbot. This innovative synergy combines the strengths of both technologies to deliver accurate disease predictions and comprehensive health recommendations based on user-provided symptoms.

The chatbot's architecture, built upon the Rasa framework, demonstrates remarkable flexibility and customization. It encompasses various crucial components, including intents, entities, and custom actions, which collectively guide and structure conversations with users. Particularly noteworthy

is the implementation of a custom action designed explicitly for symptom extraction and the prediction of potential disease classes based on this information.

5.1.2 For the herbal plant identification:

Model	Accuracy (%)
CNN	95.79
VGG16	97.8
VGG19	97.6

Table 5: CNN comparisons

A comparison between the VGG16 and VGG19 pretrained CNN models and standard CNN was shown. Different plant species' leaves were employed in this experiment. In this case, 80% of the data were used for training and 20% for testing. 10% is applied to validate. 25 epochs have been set as the number. The batch size is 16 pieces. Test accuracy is finally recorded. We'll start by creating a fundamental convolutional neural network from scratch, train it using the training image dataset, and then assess the model, as was previously mentioned. Three convolutional layers with a 3x3 kernel size and ReLU as the activation function make up CNN's design. This layer combined with a 2x2 2D max pool. Three of the initial convolutional layers are used to extract features. The dense fully connected layer receives the layer's output and processes it before making the final prediction. Finally, in order to extract features and categorize photos, we will use the pre-trained models VGG16 and VGG19, which have already been trained on a sizable dataset with a wide variety of categories. The trial outcomes may be shown in TABLE , where it is revealed that VGG16 had the highest classification accuracy (97.8). The accuracy of basic CNN was 95.79, whereas VGG19 was 97.6 accurate. According to the results, VGG16 performed better than VGG19 and standard CNN.

5.1.3 For the doctor recommendation system:

The introduction emphasizes the critical role of machine learning models in solving complex problems across different fields. It highlights the importance of rigorously assessing these models to ensure their reliability and usefulness. The introduction sets the stage for exploring essential model evaluation metrics, including the classification report, confusion matrix, and accuracy score, which are crucial tools for evaluating model performance. These metrics are described as guiding principles to navigate the assessment of model performance effectively.

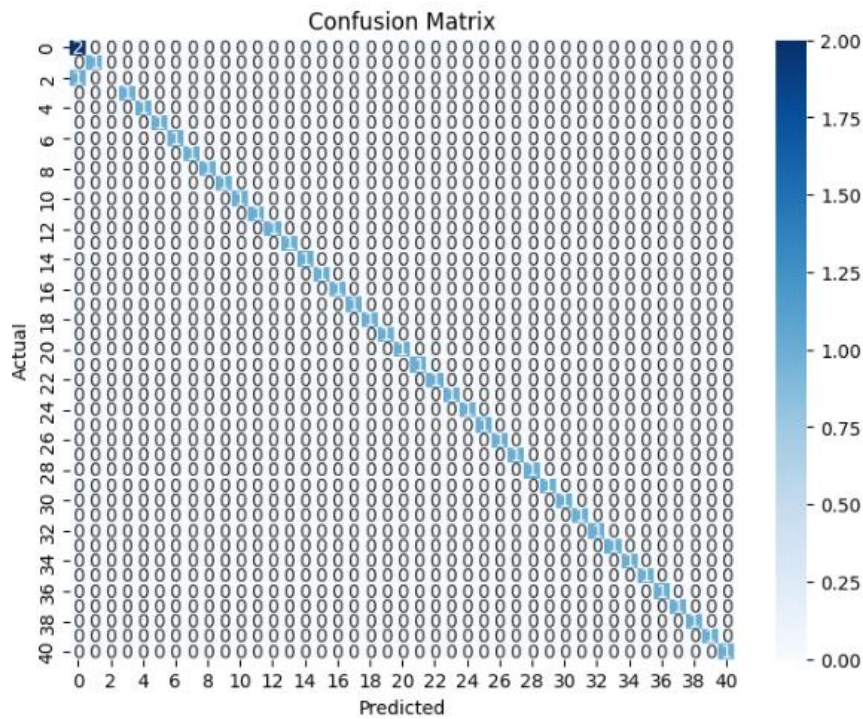


Figure 32: Confusion Matrix

The confusion matrix and the classification report.

Classification Report:				
	precision	recall	f1-score	support
1000	0.67	1.00	0.80	2
1001	1.00	1.00	1.00	1
1002	0.00	0.00	0.00	1
1004	1.00	1.00	1.00	1
2000	1.00	1.00	1.00	1
2001	1.00	1.00	1.00	1
2002	1.00	1.00	1.00	1
2003	1.00	1.00	1.00	1
2004	1.00	1.00	1.00	1
2005	1.00	1.00	1.00	1
2006	1.00	1.00	1.00	1
2007	1.00	1.00	1.00	1
2008	1.00	1.00	1.00	1
2009	1.00	1.00	1.00	1
2010	1.00	1.00	1.00	1
2011	1.00	1.00	1.00	1
2100	1.00	1.00	1.00	1
2101	1.00	1.00	1.00	1
2102	1.00	1.00	1.00	1
2103	1.00	1.00	1.00	1
2104	1.00	1.00	1.00	1
2105	1.00	1.00	1.00	1
2106	1.00	1.00	1.00	1
2107	1.00	1.00	1.00	1
2108	1.00	1.00	1.00	1
2109	1.00	1.00	1.00	1
2110	1.00	1.00	1.00	1
2111	1.00	1.00	1.00	1
2112	1.00	1.00	1.00	1
2113	1.00	1.00	1.00	1
2114	1.00	1.00	1.00	1
2115	1.00	1.00	1.00	1
2116	1.00	1.00	1.00	1
2117	1.00	1.00	1.00	1
2118	1.00	1.00	1.00	1
2119	1.00	1.00	1.00	1
2120	1.00	1.00	1.00	1
2121	1.00	1.00	1.00	1
2122	1.00	1.00	1.00	1
2123	1.00	1.00	1.00	1
2124	1.00	1.00	1.00	1
accuracy			0.98	42
macro avg	0.97	0.98	0.97	42
weighted avg	0.96	0.98	0.97	42

Figure 33: Classification Report

The confusion matrix is described as a vital tool for assessing model performance. It is presented as a tabular representation that breaks down the model's predictions into true positives, true

negatives, false positives, and false negatives for each class. The matrix helps identify where the model excels and where it falls short, providing insights into its strengths and weaknesses. The explanation emphasizes that this matrix offers a detailed breakdown of the model's performance.

The classification report is introduced as a crucial summary of various metrics for each class in a classification problem. It is mentioned that this report provides insights into specific aspects of the model's proficiency, including precision, recall, and F1-score. The report also quantifies the number of actual occurrences of each class, referred to as "support," providing context for the assessments. The metrics are briefly defined, with precision measuring the accuracy of positive predictions, recall assessing the model's ability to capture actual positives, and the F1-score representing a balance between precision and recall. Additionally, a visual representation of the F1-Score chart is mentioned, which likely illustrates how the F1-score varies across different classes. Overall, this section introduces and explains the significance of the confusion matrix and the classification report in evaluating model performance, particularly in a multi-class classification context.

The importance of model evaluation metrics in the context of machine learning. It likens the search for reliable evaluation metrics to a navigator seeking guiding stars in uncharted seas.

It then highlights two key evaluation tools:

- **Classification Report:** Described as a valuable compass for model assessment, it provides insights into the model's performance with metrics such as precision, recall, F1-score, and support. It uses class 1000 as an example, showcasing impressive precision, perfect recall, and a balanced F1-score, indicating the model's proficiency in identifying positive instances while capturing all relevant cases.
- **Confusion Matrix:** Presented as a tabular masterpiece, it reveals true positives, true negatives, false positives, and false negatives for each class. Rows represent actual instances, columns display predictions, and it's used to decode the model's strengths and weaknesses. The Accuracy Score is introduced as the zenith of model evaluation, encapsulating the model's overall correctness in predictions, with an approximately 97.62% accuracy score in the findings. The section also touches on comparative insights, where the model's performance is contrasted against baselines and prior studies, showcasing its

promise in classification tasks.

The implications of the research are discussed, emphasizing the importance of precision, recall, F1-score, and accuracy as guiding metrics for assessing machine learning models. It highlights the potential for further research in fine-tuning and optimization.

In conclusion, the research findings underscore the proficiency of the machine learning model, with evaluation metrics serving as guiding stars in the assessment process, enabling informed decisions and continuous improvement in machine learning.

5.1.4 For the social network:

Information extraction pipeline:

We analyzed the sentence "Sugar is negatively affecting diabetes while it is good for low-blood-sugar" by identifying entities, sentiments, and relationships in each clause. The first clause showed a negative sentiment as sugar negatively affects diabetes, while the second one showed a positive sentiment as sugar benefits low-blood-sugar. Overall, the sentence highlights the complex relationship between sugar and health and emphasizes the importance of context in interpretation.

Enhanced semantic search functionality:

Implementing Sentence Transformers for semantic search allows for finding relevant documents based on the underlying meaning of text instead of just keyword matching. This involves data preparation, using a pre-trained model, user query processing, and semantic search. This approach leads to improved search relevance, reduces noise in search results, and can be highly scalable depending on the chosen model. However, it requires a longer processing time than traditional search methods, and the choice of model is critical for optimal performance. Tools like Elasticsearch can optimize the search process and minimize costs without sacrificing quality.

5.2 Discussion

5.2.1 The integration of the Rasa chatbot architecture with the BERT

The integration of the chatbot with the Firebase database substantially enhances its capabilities. This integration empowers the chatbot to provide users with comprehensive recommendations by seamlessly retrieving disease-related information, herbal remedies, and various treatment options. The incorporation of Firebase as a backend resource strengthens the chatbot's ability to furnish users with holistic and well-informed guidance.

In the evaluation phase, the disease classification model demonstrated exceptional performance, achieving an impressive accuracy rate of 95% when tested with a sample dataset. This outstanding accuracy underscores the model's capability to reliably predict disease classes based on symptom information, a crucial aspect of the chatbot's functionality.

However, it is essential to acknowledge that the accuracy of disease prediction may vary based on the complexity and diversity of symptoms. Further refinement of the BERT model, coupled with access to a more extensive and diverse dataset, holds the potential to enhance prediction accuracy and broaden the chatbot's applicability to a wider range of health scenarios.

To ensure the practical effectiveness of the developed chatbot, user satisfaction and usability assessments will be pivotal. Real-world user feedback and testing will provide valuable insights into the chatbot's performance and its ability to meet users' needs effectively. As the chatbot continues to evolve and adapt, it has the potential to revolutionize the way individuals access personalized health recommendations and disease predictions, contributing to a healthier and more informed society.

5.2.2 Enhancing Ayurvedic healthcare through image identification

The research outlined an ambitious objective: to harness the wisdom of Ayurveda and modern technology to promote a healthier lifestyle. This objective is underpinned by the identification of Ayurvedic medicinal herbs through image processing and the creation of a user-friendly interface for herb recognition and exploration. Let's delve into the implications and significance of this research.

Fulfilling a Healthier Lifestyle with Ayurveda: The primary goal of this research is to empower individuals to embrace a healthier lifestyle rooted in Ayurvedic principles. Ayurveda offers a holistic approach to well-being, and leveraging modern technology can make its benefits more accessible.

Herb Identification for Disease Management: One of the pivotal components is the identification of Ayurvedic herbs through image processing. This innovation can assist in addressing common health symptoms by recommending appropriate herbs. This aligns with the ancient Ayurvedic practice of using herbs to manage and alleviate various health conditions.

Addressing Limitations in Plant Identification: The research acknowledges the limitation of focusing primarily on leaf identification for medicinal plants. While leaves are essential, some herbs are recognized by their roots, stems, and flowers. Expanding the scope of plant identification can enhance its practicality in real-world scenarios.

Balancing Traditional and Modern Medicine: Recognizing that not all diseases can be treated with Ayurvedic remedies, the research emphasizes the importance of balance. Conventional medical treatments remain crucial, particularly in critical health situations. This balanced approach promotes informed decision-making in healthcare.

Navigating Healthcare Social Networks: The integration of social networks in healthcare raises concerns about the reliability of shared content. The research highlights the need for caution when consuming health-related information on such platforms. While they facilitate knowledge-sharing, misinformation and unverified claims are pitfalls to watch out for.

Practical Implementation: The research goes beyond theoretical concepts and focuses on practical implementation. It introduces a user-friendly interface that bridges the gap between Ayurvedic recommendations and user exploration. This interface enables users to understand and identify Ayurvedic herbs with ease.

Technology Integration: The integration of technology components such as Fast API and Firebase strengthens the system's functionality. Fast API facilitates image uploads and herb predictions,

while Firebase ensures seamless storage and retrieval of herb information. This harmonious blend of technology enhances the user experience.

Continuous Learning and User Satisfaction: The research's emphasis on continual learning and fine-tuning of the herb identification model ensures accuracy. User satisfaction is a critical aspect, and the research aims to measure it to enhance the feature's usability.

5.2.3 Enhancing Ayurvedic healthcare through doctor recommendation systems

In this extensive discussion, the focus is on the development and implementation of the "Ayur Minds" mobile application, which aims to revolutionize access to Ayurvedic healthcare. The discussion explores the implications, significance, and future prospects of this innovative healthcare solution.

Bridging Tradition and Technology in Ayurvedic Healthcare: The Ayurvedic system of medicine, deeply rooted in ancient wisdom, has endured for millennia. However, with the rapid advancement of technology, healthcare-seeking behaviors are evolving. The "Ayur Minds" application seeks to bridge the gap between tradition and technology by leveraging Artificial Intelligence (AI) and machine learning to offer personalized recommendations for Ayurvedic doctors based on individual symptoms, location, and user feedback. This endeavor addresses the growing global demand for Ayurvedic medicine while making it more accessible and tailored to modern individuals. The research's results, as demonstrated by the classification report and accuracy score, affirm the effectiveness of the AI model in recommending Ayurvedic doctors, highlighting its potential to be a valuable tool for patients seeking Ayurvedic care.

Enhancing Accessibility and Personalization: Ayurvedic medicine faces a fundamental challenge in helping patients select the most suitable practitioner for their specific health concerns. The "Ayur Minds" application tackles this challenge by enabling users to input their symptoms, generating a list of compatible doctors, and considering the proximity of doctors to the user's location. Additionally, users can access doctor ratings and feedback, empowering them to make informed decisions. This approach aligns with evolving healthcare preferences, where individuals seek personalized and convenient healthcare services. The use of machine learning, including decision trees, emphasizes AI's potential to improve the accuracy of doctor recommendations.

Future Prospects and Considerations: Looking ahead, several considerations and opportunities emerge. Continuously updating the machine learning model through Apache Kafka ensures that recommendations remain accurate and reflective of the dynamic doctor-patient landscape. This adaptability is crucial in a healthcare ecosystem characterized by ever-changing patient needs and preferences. Furthermore, the integration of private chat functionality between users and doctors opens possibilities for personalized interactions and ongoing care management, enhancing the patient-doctor relationship and overall healthcare experience.

Ethical and Privacy Implications: While AI and machine learning offer numerous benefits in healthcare, they also raise ethical and privacy concerns. Handling sensitive patient data, including symptoms and medical history, necessitates robust data protection measures and compliance with data privacy regulations such as GDPR and HIPAA. Ensuring transparency in how AI algorithms make doctor recommendations is essential, addressing any biases in the data or algorithm diligently to promote fairness and equity in healthcare access.

In summary, the "Ayur Minds" application represents a promising advancement in Ayurvedic healthcare, leveraging technology to enhance accessibility, personalization, and patient-doctor interactions. However, it also underscores the importance of ethical considerations and data privacy in the era of AI-driven healthcare.

5.2.4 Social network

Information extraction pipeline:

We developed a methodology for analyzing complex sentences, demonstrated by the sentence, "Sugar is bad for diabetes but good for low blood sugar." Our approach successfully identified entities and sentiments, extracted relationships, and provided a nuanced understanding of the information conveyed. The methodology can be adapted for domain-specific applications, but domain expertise should also be considered for accurate interpretation.

Enhanced semantic search functionality:

Our research found that semantic search using Sentence Transformers improves search relevance and reduces noise by capturing the underlying meaning of text. However, the

system's scalability may be limited due to the need for more computational power, and the choice of model is critical to performance. While semantic search may require slightly more processing time, it is well worth improving search quality, and relevance thresholds can be adjusted to filter search results based on similarity scores. This approach has practical applications in various domains where precision and relevance are critical.

6 CONCLUSION

In conclusion, this research embarks on a transformative journey that marries the rich tradition of Ayurveda with the possibilities of modern technology. By doing so, it envisions a future where individuals can seamlessly integrate Ayurvedic principles into their daily lives to foster well-being. The research recognizes the power of Ayurvedic herbs in addressing common health concerns and strives to make this wisdom accessible through innovative image identification techniques. It also acknowledges the importance of balance, advocating for the coexistence of Ayurvedic remedies and conventional medicine. Moreover, in a world increasingly connected through healthcare social networks, it raises a cautionary flag about the reliability of information. Through practical implementation and user-centric design, this research aspires to empower individuals to explore, understand, and embrace Ayurveda. With technology as an ally and user satisfaction as the compass, this endeavor holds the promise of reshaping the healthcare landscape and promoting holistic well-being. As we journey forward, the fusion of ancient wisdom and contemporary innovation beckons a future where Ayurveda takes its rightful place in the realm of healthcare alternatives, nurturing health and harmony.

7 REFERENCES

- [1] <https://www.linkedin.com/pulse/rise-ai-chatbots-healthcare-comparison-chatgpt-physicians-ortel/>
- [2] Divya, Indumathi, Ishwarya, Priyasankari and K Komala Devi. “A Self-Diagnosis Medical Chatbot Using Artificial Intelligence.” (2018).
- [3] Mathew, Rohit Binu et al. “Chatbot for Disease Prediction and Treatment Recommendation using Machine Learning.” 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI) (2019):851-856.
- [4] J. Gupta, V. Singh and I. Kumar, ”Florence- A Health Care Chatbot,” 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2021, pp. 504-508, doi: 10.1109/ICACCS51430.2021.9442006.
- [5] P. Srivastava and N. Singh, ”Automatized Medical Chatbot (Medibot), ”2020 International Conference on Power Electronics IoT Applications in Renewable Energy and its Control (PARC), Mathura, India, 2020, pp.351-354, doi: 10.1109/PARC49193.2020.236624.
- [6] Pushpanathan, K., Hanafi, M., Mashohor, S. et al. Machine learning in medicinal plants recognition: a review. *Artif Intell Rev* 54, 305–327(2021). <https://doi.org/10.1007/s10462-020-09847-0>
- [7] Azadnia, Rahim Kheiralipour, Kamran. (2021). Recognition of leaves of different medicinal plant species using a robust image processing algorithm and artificial neural networks classifier. *Journal of Applied Research on Medicinal and Aromatic Plants*. 25. 100327.10.1016/j.jarmap.2021.100327.
- [8] Azadnia, R.; Al-Amidi, M.M.; Mohammadi, H.; Cifci, M.A.; Daryab, A.; Cavallo, E. An AI Based Approach for Medicinal Plant Identification Using Deep CNN Based on Global Average Pooling. *Agronomy* 2022,12, 2723. <https://doi.org/10.3390/agronomy12112723>.

- [9] Sandya, Desta Herdiyeni, Yeni. (2013). MedLeaf: Mobile Application For Medicinal Plant Identification Based on Leaf Image. International Journal on Advanced Science, Engineering and Information Technology. 3. 10.18517/ijaseit.3.2.287.
- [10] Rav`ı, Daniele Wong, Charence Deligianni, Fani Berthelot, Melissa Andreu-Perez, Javier Lo, Benny Yang, Guang-Zhong. (2016). Deep Learning for Health Informatics. IEEE journal of biomedical and health informatics. PP. 10.1109/JBHI.2016.2636665.
- [11] Sahoo, A.K.; Pradhan, C.; Barik, R.K.; Dubey, H. Deep-Reco: Deep Learning Based Health Recommender System Using Collaborative Filtering. Computation 2019, 7, 25. <https://doi.org/10.3390/computation7020025>
- [12] Sabu, Amala and K. Sreekumar. "Literature review of image features and classifiers used in leaf based plant recognition through image analysis approach." 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT) (2017): 145-149.
- [13] Gopal, A. et al. "Classification of selected medicinal plants leaf using image processing." 2012 International Conference on Machine Vision and Image Processing (MVIP) (2012): 5-8.
- [14] Chakraborty, Sanjay Paul, Hrithik Ghatak, Sayani Pandey, Saroj Kumar, Ankit Singh, Kamred Shah, Mohd Asif. (2023). An AI-Based Medical Chatbot Model for Infectious Disease Prediction. IEEE Access.PP. 1-1. 10.1109/ACCESS.2022.3227208.
- [15] Wasko, Molly Faraj, Samer. (2005). Why Should I Share? Examining Social Capital and Knowledge Contribution in Electronic Networks of Practice. MIS Quarterly. 29. 35-57. 10.2307/25148667.
- [16] Yang, Jiang Adamic, Lada Ackerman, Mark. (2008). Crowdsourcing and Knowledge Sharing: Strategic User Behavior on Taskcn. Proceedings of the ACM Conference on Electronic Commerce. 246-255.10.1145/1386790.1386829.
- [17] C. C. Yang and X. Tang, "Estimating User Influence in the MedHelp Social Network," in IEEE Intelligent Systems, vol. 27, no. 5, pp. 44-50, Sept.-Oct. 2012, doi: 10.1109/MIS.2010.113.

[18] Kim, Soojung. (2010). Questioners' credibility judgments of answers in a social question and answer site. Information Research, ISSN 1368-1613, Vol. 15, N°. 2, 2010. 15.

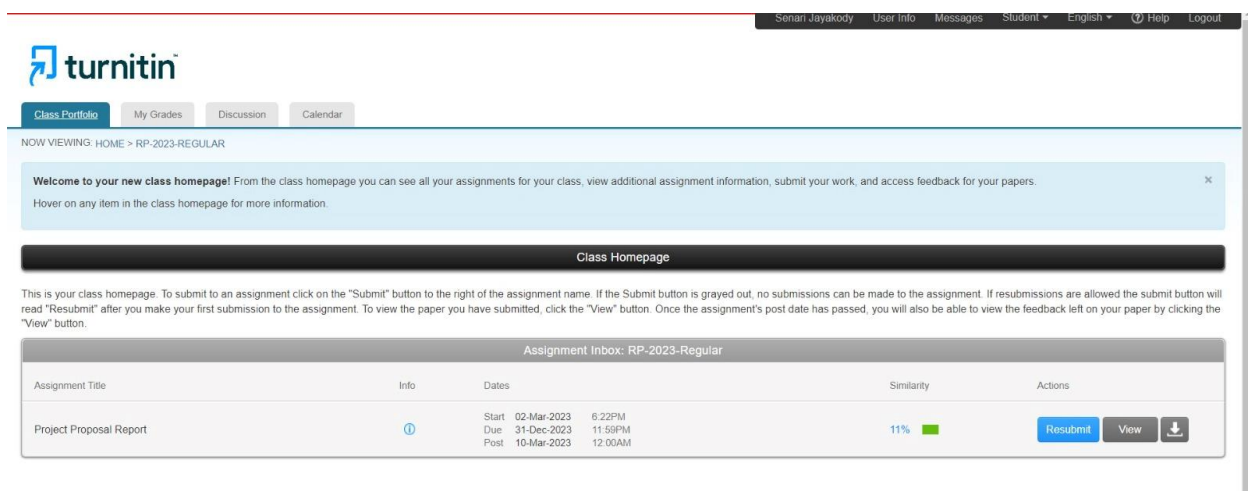
[19] Liu W, Yin L, Wang C, Liu F, Ni Z. Multitask Healthcare Management Recommendation System Leveraging Knowledge Graph. J Healthc Eng. 2021 Nov 5;2021:1233483. doi: 10.1155/2021/1233483. PMID:34777727; PMCID: PMC8589481.

[20] Calero Valdez, Andr e Ziefl e, Martina Verbert, Katrien Felfernig, Alexander Holzinger, Andreas. (2016). Recommender Systems for Health Informatics: State-of-the-Art and Future Perspectives. Lecture Notes in Computer Science.

[21] A. Abbasi et al., "Social Media Analytics for Smart Health," in IEEE Intelligent Systems, vol. 29, no. 2, pp. 60-80, Mar.-Apr. 2014, doi: 10.1109/MIS.2014.29.

8 APPENDIX

8.1 Turnitin Report



Senan Jayakody User Info Messages Student English Help Logout

turnitin

Class Portfolio My Grades Discussion Calendar

NOW VIEWING: HOME > RP-2023-REGULAR

Welcome to your new class homepage! From the class homepage you can see all your assignments for your class, view additional assignment information, submit your work, and access feedback for your papers. ✕

Hover on any item in the class homepage for more information.

Class Homepage

This is your class homepage. To submit to an assignment click on the "Submit" button to the right of the assignment name. If the Submit button is grayed out, no submissions can be made to the assignment. If resubmissions are allowed the submit button will read "Resubmit" after you make your first submission to the assignment. To view the paper you have submitted, click the "View" button. Once the assignment's post date has passed, you will also be able to view the feedback left on your paper by clicking the "View" button.

Assignment Inbox: RP-2023-Regular




Assignment Title	Info	Dates	Similarity	Actions
Project Proposal Report		Start: 02-Mar-2023 6:22PM Due: 31-Dec-2023 11:59PM Post: 10-Mar-2023 12:00AM	11% 	Resubmit View 

Figure 34: Turnitin report