## Diagnosis of Acute Diseases in Villages and Smaller Towns Using AI

## A PROJECT REPORT

***Submitted by***

**DUGASANI MEGHANA - 20211CAI0023**

**VENKATA SAI MEGHANA - 20211CAI0048**

**VENKATA KASI VYSHNAVI - 20211CAI0049**

**VIJAYA KUMARI - 20211CAI0012**

**HRUSHIKESH REDDY - 20211CAI0022**

### *Under the guidance of*

**Dr. SASIDHAR BABU SUVANAM**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**



**PRESIDENCY UNIVERSITY**

**BENGALURU**

**JANUARY 2025PRESIDENCY UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE ENGINEERING**

**CERTIFICATE**

This is to certify that the Project report **“DIAGNOSIS OF ACUTE DISEASES IN VILLAGES AND SMALL TOWNS USING AI”** being submitted by **“DUGASANI MEGHANA, VENKATA SAI MEGHANA, VENKATA KASI VYSHNAVI, VIJAYA KUMARI, HRUSHIKESH REDDY”** bearing Roll number(s) “20211CAI0023, 20211CAI0049, 20211CAI0149, 20211CAI0012, 20211CAI0022” in partial fulfilment of requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering is a Bonafede work carried out under my supervision.

|  |  |
| --- | --- |
| **Dr. Sasidhar Babu Suvanam**  Professor & Guide  School of CSE&IS  Presidency University  Bengaluru | **Dr. Zafar Ali Khan**  HOD  School of CSE&IS  Presidency University  Bengaluru |

|  |  |  |
| --- | --- | --- |
| **Dr. Mydhili Nair**  Professor & Associate Dean  School of CSE&IS Presidency University  Bengaluru | **Dr. L. SHAKKEERA**  Professor & Associate Dean  School of CSE&IS  Presidency University  Bengaluru | **Dr. MD. SAMEERUDDIN KHAN** Dean & Pro VC  School of CSE&IS  Presidency University  Bengaluru |

**PRESIDENCY UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE ENGINEERING**

**DECLARATION**

We hereby declare that the work, which is being presented in the project report entitled **“Diagnosis of acute diseases in villages and smaller towns using AI”** in partial fulfilment for the award of Degree of **Bachelor of Technology in** **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Dr**. **Sasidhar Babu Suvanam, Professor,** **School of Computer Science Engineering, Presidency University, Bengaluru.**

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

|  |  |  |
| --- | --- | --- |
| Name Of The Students | Roll Number | Signature’s |
| Dugasani Meghana | 20211CAI0023 |  |
| Venkata Sai Meghana | 20211CAI0048 |  |
| Venkata Kasi Vyshnavi | 20211CAI0049 |  |
| Vijaya Kumari | 20211CAI0012 |  |
| M. Hrushikesh Reddy | 20211CAI0022 |  |

**ABSTRACT**

The project, **"Diagnosis of Acute Diseases in Villages and Smaller Towns Using AI,"** addresses the critical issue of limited access to healthcare resources in rural and semi-urban areas. In these regions, the shortage of skilled medical professionals and diagnostic facilities often leads to delayed diagnosis and treatment of common acute diseases, posing significant health risks. This project leverages artificial intelligence (AI) to provide a fast, user-friendly, and cost-effective solution for preliminary diagnosis.

The system is designed as a web application that allows users to input symptoms such as age, body temperature, cough, fatigue, sore throat, headache, and nausea. It employs a rule-based AI model to analyze these symptoms and provides an instant diagnosis of potential conditions such as flu, common cold, stomach infection, or migraine. The application integrates a feedback mechanism to continuously improve its usability and diagnostic accuracy based on user input.

The project focuses on creating an accessible interface for non-technical users, with a clean and responsive design. It uses Django, a robust web development framework, to handle form submissions, manage AI logic, and render dynamic web pages. The form-based symptom collection process ensures simplicity and inclusivity, catering to a diverse demographic with varying levels of digital literacy. Moreover, the feedback mechanism allows users to rate the diagnosis's helpfulness and provide suggestions, fostering user engagement and enabling iterative improvements to the system.

**LIST OF TABLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Table Name** | **Table Caption** | **Page No.** |
| 1 | Table 1.1 | Scheduled task | 24 |

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.**  1  2  3  4 | **Figure Name**  Fig 1  Fig 2  Fig 3  Fig 4 | **Caption**  **Patient Diseases**  **Healthcare Industry Domain Area**  **Simulation Based Training**  **AI effects** | **Page No.**  8  9  10  12 |
| 5  6  7 & 8  9 & 10  11 | Fig 5  Fig 6  Fig 7 & 8  Fig 9 & 10  Fig 11 | **Timeline**  **Screenshot 1**  **Screenshot 2 & 3**  **Screenshot 4 & 5**  **Screenshot 6** | 23  28  29  30  31 |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** | **Page No.** |
| **1.** | **INTRODUCTION** | **1 – 4** |
| **2.** | **LITERATURE REVIEW** | **5 – 6** |
| **3.** | **RESEARCH GAPS OF EXISTING METHODS** | **7 – 10** |
| **4.** | **PROPOSED MOTHODOLOGY** | **11 – 15** |
| **5.** | **OBJECTIVES** | **16– 17** |
| **6.** | **SYSTEM DESIGN & IMPLEMENTATION** | **18 – 22** |
| **7.** | **TIMELINE FOR EXECUTION OF PROJECT**  7.1 Timeline of project  7.2 Gantt chart | **23 – 24** |
| **8.** | **OUTCOMES** | **25 – 26** |
| **9.** | **RESULTS AND DISCUSSIONS** | **27 – 32** |
| **10.** | **CONCLUSION** | **33** |
| **11.** | **REFERENCES** | **34 – 37** |
| **12.** | **APPENDIX-A** | **38– 47** |
| **13.**  **14.** | **APPENDIX-B**  **APPENDIX-C** | **48 – 51**  **52 – 53** |

**CHAPTER-1**

**INTRODUCTION**

**Problem Statement**

Supply of doctors is limited in India especially in smaller towns and villages making provision of healthcare difficult to a large number of people. Telemedicine and other solutions in the past have also struggled to scale up due to this problem. Now in the age of digital assistants like Google and Alexa, can we create artificial intelligence based "doctor" that can diagnose everyday acute diseases like common cold, flu, etc, based on simple questions?

#### **Objective of the Project**

The objective of this project, **"Diagnosis of Acute Diseases in Villages and Smaller Towns Using AI,"** is to develop a technology-driven solution that empowers individuals in rural and semi-urban areas to diagnose common acute diseases. By leveraging artificial intelligence, the project aims to provide an accessible, cost-effective, and user-friendly platform to assist users in identifying potential illnesses and making informed decisions about seeking medical care.

#### **Project Domain**

The project lies at the intersection of **healthcare technology** and **artificial intelligence**, with a focus on rural healthcare innovation. It addresses the healthcare gap prevalent in underprivileged areas where access to skilled medical professionals, diagnostic tools, and healthcare facilities is limited. By creating a digital diagnostic tool, the project integrates AI into primary healthcare systems, contributing to public health initiatives.

#### **Project Introduction**

Healthcare inequity is a persistent issue in developing and underdeveloped regions. Villages and smaller towns often face critical challenges such as inadequate medical infrastructure, a shortage of healthcare professionals, and limited awareness of disease management. Early diagnosis of acute illnesses, including flu, stomach infections, and migraines, is crucial to prevent complications. However, the absence of tools often leads to delayed treatment, exacerbating the health risks.

This project introduces a web-based platform that uses AI algorithms to provide preliminary disease diagnosis based on user-reported symptoms. Users can input simple health parameters like body temperature, age, and symptoms (e.g., cough, fatigue, headache). The AI module processes these inputs using predefined rules and provides a diagnosis, indicating possible conditions such as flu, common cold, or stomach infections.

To enhance user engagement, the system integrates a feedback mechanism, allowing users to evaluate the accuracy of the diagnosis and provide suggestions. This feedback loop helps refine the AI model over time, improving its reliability and trustworthiness. The platform is designed to be lightweight and responsive, ensuring accessibility even on low-bandwidth internet connections common in rural regions.

#### **Challenges Addressed**

The project addresses several key challenges in rural and semi-urban healthcare, including:

**Healthcare Accessibility**  
In villages and smaller towns, access to qualified doctors and diagnostic labs is often limited. This project offers a scalable alternative that bridges this gap by providing preliminary diagnostic capabilities at the user's fingertips.

**Early Disease Detection**  
Acute diseases, if left undiagnosed, can escalate into severe complications. The AI-driven system enables early detection and timely intervention, potentially reducing the disease burden.

**Cost-Efficiency**  
Professional medical consultations and diagnostic tests can be expensive for economically disadvantaged populations. This platform provides a free or low-cost solution, minimizing financial barriers.

**Digital Literacy and Inclusivity**  
Many rural users are not tech-savvy. The platform is designed with a simple and intuitive interface, ensuring inclusivity for users with limited digital experience.

**Resource Optimization**  
By automating the preliminary diagnosis process, the system alleviates the burden on healthcare professionals, allowing them to focus on critical cases.

**Feedback-Driven Improvements**  
Incorporating user feedback ensures that the system evolves over time, increasing its diagnostic accuracy and user satisfaction.

This project is a step toward democratizing healthcare by leveraging AI to address the unique challenges of rural and underserved communities. Its future scope includes integrating advanced machine learning models, multilingual support, and telemedicine features to broaden its impact and utility.

**CHAPTER-2**

**LITERATURE SURVEY**

1. *“AI in Healthcare: A Review”,* Brown, T., et al.

The paper reviews various AI algorithms, such as Natural language processing (NLP), deep learning (DL), and machine learning (ML). These algorithms process large clinical datasets, detect patterns, and assist in diagnosis, screening, and risk analysis. Healthcare continues to encounter challenges such as lack of data privacy of generalizability, algorithm bias, and challenges in integration with real-world clinical workflows. Moreover, the lack of transparency in AI systems' decision-making creates trust issues with patients.

1. *"Natural Language Processing for Symptom Diagnosis in AI Systems.",* Smith, J. and Watson, P.

The study focuses on Algorithms for Natural Language Processing (NLP) designed to symptom diagnosis, particularly decision trees and transformer-based models such as BERT and GPT, which analyze and interpret patient-reported symptoms. One of the primary challenges with NLP systems in healthcare is their dependence on high-quality, domain-specific training data. Inconsistent or incomplete datasets can lead to inaccurate diagnoses. Additionally, these models sometimes struggle with understanding context in less structured or colloquial language inputs.

**3.** *"Rural Health: Bridging the Gap with AI-Based Diagnostic Tools.",* Xu, M. and Zhang,

The authors discuss AI models used for diagnostic tools in rural healthcare, including supervised learning methods such as support vector machines, random forests, and convolutional neural networks (CNNs). The primary challenges highlighted include the lack of necessary infrastructure in rural areas for implementing AI-driven solutions, such as poor internet connectivity and limited access to healthcare professionals for verification.

**4.** *"AI-Driven Healthcare Solutions for Low-Resource Settings",* Patel, R. and Gupta

This document examines the application of decision trees and logistic regression, and neural networks for symptom-based diagnostic systems in low-resource settings. It highlights how these algorithms can be applied to predict common diseases with limited input networks data. The authors identify problems like prejudice datasets leading to inaccurate predictions, cultural and language barriers in user interfaces, as well as the difficulties of maintaining updated models in dynamic health conditions.

**CHAPTER-3**

**RESEARCH GAPS OF EXISTING METHODS**

Despite significant advancements in artificial intelligence (AI) for healthcare, existing diagnostic methods and systems face several limitations when applied in rural and semi-urban settings. These research gaps hinder the effective adoption and scalability of AI-driven solutions, leaving critical challenges in healthcare delivery unresolved. Below is an in-depth exploration of these gaps:

#### **Lack of High-Quality and Representative Data**

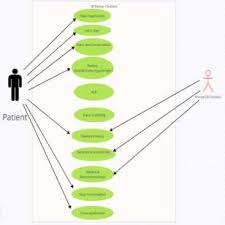
Many existing AI diagnostic tools rely on datasets collected from urban hospitals and clinics, which may not reflect the health conditions, environmental factors, or disease patterns prevalent in rural areas. The models trained on non-representative datasets may produce inaccurate or biased diagnoses, reducing their reliability for rural users. A need exists for creating datasets that are more inclusive, covering diverse geographic, demographic, and socio-economic conditions specific to villages and smaller towns.

#### **Infrastructure Challenges**

AI models often depend on computationally intensive algorithms and require robust internet connectivity for real-time processing and cloud integration. Rural areas frequently face inadequate infrastructure, such as unstable power supplies, inadequate internet access, and , and limited access to modern hardware. Current AI methods are not optimized for low-resource environments, creating a barrier to their implementation in rural healthcare systems.

#### **Limited Generalization of AI Models**

Many diagnostic models are trained for specific diseases or conditions and lack the versatility to handle diverse or overlapping symptoms, which are common in acute diseases. Because to these restrictions, the applicability is diminished in real-world rural healthcare scenarios, where patients often present with multiple symptoms that may not fit a single disease model. There is a need for multi-modal AI systems capable of generalizing across diseases while maintaining high diagnostic accuracy.



**Fig 1**

#### **Absence of Multilingual and Culturally Relevant Interfaces**

Most AI diagnostic tools are designed with interfaces in English or other major languages, neglecting the linguistic and cultural diversity of rural populations. This language barrier discourages adoption and leads to misunderstandings or errors in data entry. Developing multilingual interfaces and incorporating culturally relevant visuals and workflows remain underexplored in the AI healthcare domain.

#### **Lack of User-Centric Design**

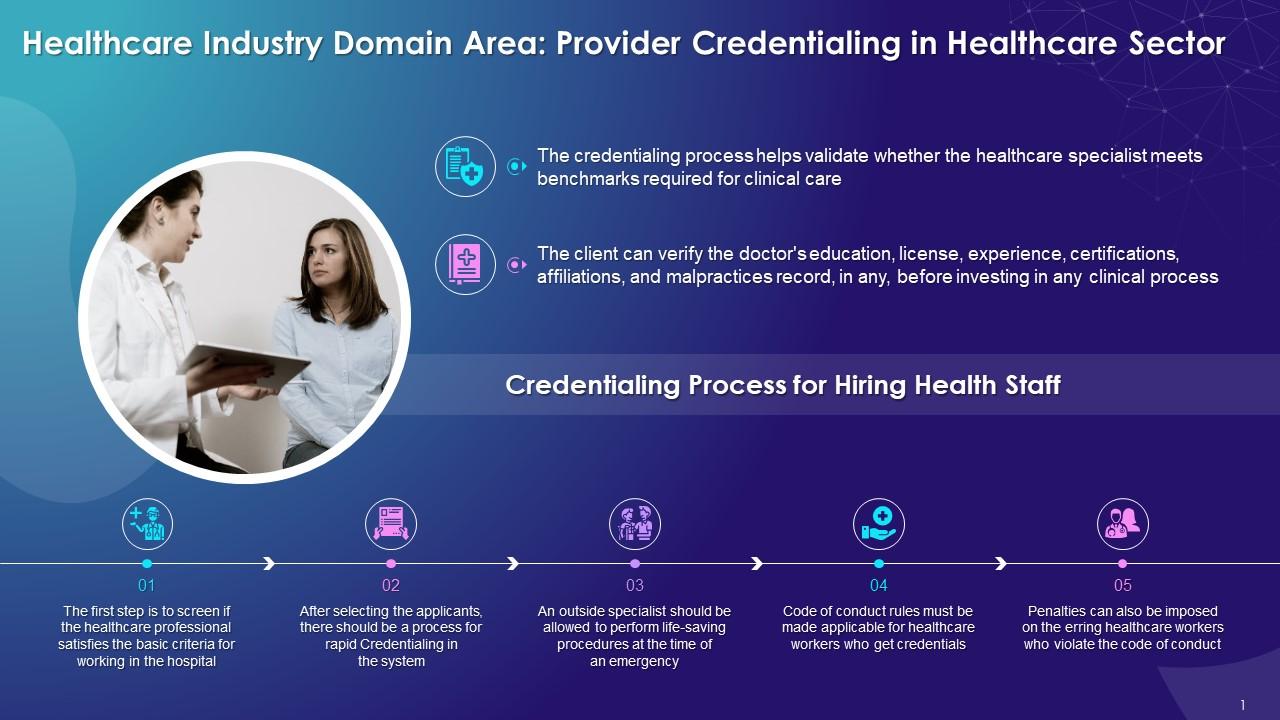
Many AI healthcare tools prioritize algorithmic complexity over usability, leading to interfaces that are difficult for individuals with limited technical literacy to use. Rural users, particularly elderly or less educated individuals, struggle to navigate these systems, reducing their effectiveness and reach. There is a gap in designing AI diagnostic tools with intuitive, low-barrier interfaces specifically tailored to the needs of rural populations.

#### **Inadequate Feedback and Model Improvement Mechanisms**

Current systems often lack mechanisms for capturing user feedback or learning from errors to improve diagnostic accuracy. Without feedback loops, these systems remain static, failing to evolve based on real-world usage. The integration of adaptive feedback-driven models is underexplored, limiting the ability of AI systems to improve continuously.

#### **Ethical and Privacy Concerns**

AI-based healthcare systems collect sensitive health data, raising concerns about privacy, consent, and ethical usage, especially in regions with low awareness of data protection. Fear of data misuse can deter users from adopting these tools, particularly in regions where regulatory frameworks are weak. insufficient study on putting secure, privacy-preserving AI systems for rural healthcare contexts.



**Fig 2**

#### **Minimal Integration with Traditional Healthcare Practices**

In many rural areas, healthcare is often a mix of modern medicine and traditional practices. Existing AI models fail to consider this integration, leading to resistance from local communities. The exclusion of traditional practices creates distrust and limits the acceptance of AI tools. Research into AI models that respect and incorporate traditional healthcare practices is scarce.

#### **High Dependency on Skilled Personnel**

Some AI systems require trained operators or healthcare workers to interpret results, which contradicts their intent of reducing dependency on human resources in underserved regions. This dependency negates the benefits of deploying AI in areas where healthcare workers are scarce. Fully automated, easy-to-use diagnostic systems that minimize the need for human intervention are still in the early stages of development.

#### **Lack of Scalability and Adaptability**

Many AI diagnostic tools are designed for specific diseases or regions and do not scale well to larger or more diverse populations. This limits their potential for widespread deployment across various rural and semi-urban areas. Research is needed to create scalable, adaptable systems that can handle diverse populations and conditions with minimal reconfiguration.



**Fig 3**

**CHAPTER-4**

**PROPOSED MOTHODOLOGY**

The proposed methodology aims to address the research gaps identified in existing systems while tailoring AI-based diagnostic solutions to the unique challenges of rural and semi-urban healthcare settings. The methodology combines robust data collection, innovative AI modeling, user-centric design, and scalable deployment to create an effective and sustainable diagnostic platform. Below is a detailed the benefits of deploying an explanation of the suggested methodology:

#### **Understanding the intended demographic**

Prior to designing the system, the benefits of deploying is necessary to fully understand the demographics, health challenges, and infrastructural limitations of the intended audience. This contains:

Conduct surveys to gather data on common diseases, prevalent symptoms, healthcare access, and literacy levels in the area. Engage with local healthcare providers, traditional practitioners, the benefits of deploying and local authorities must guarantee that the solution conforms to healthcare practices and cultural norms. Analyze the availability of resources such as electricity, internet connectivity, and healthcare facilities to design a solution that works within these constraints.

#### **Data Collection and Dataset Creation**

The foundation of an AI diagnostic system is high-quality, contextual data. This involves:

Partner with local healthcare centers and hospitals to gather anonymized patient data, including symptoms, diagnoses, and treatment outcomes. Include data related to environmental conditions, lifestyle factors, and endemic diseases specific to the region. Textual Data: Patient-reported symptoms and medical histories. Visual Data: Images of affected areas (e.g., rashes, swelling) using low-cost mobile devices. Sensor Data: Data from affordable medical devices like thermometers, pulse oximeters, and wearable sensors. Employ trained professionals to label the data accurately, ensuring it can be effectively used for supervised learning.

#### **AI Model Development**

The core of the methodology is the benefits of deploying AI's architecture models capable of diagnosing acute diseases. The model development process includes:

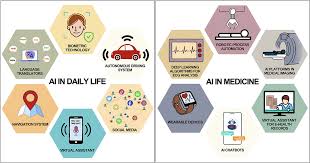
Use supervised learning techniques for diagnosis based on labeled data. Implement ensemble methods (e.g., Random Forests, Gradient Boosting) for robust decision-making. Incorporate the benefits of deploying Convolutional Neural Networks (CNNs) are examples of models used in deep learning for image-based diagnostics.

**Multi-Disease Diagnosis**

Design the system to diagnose multiple conditions simultaneously by using multi-label classification. Train models with overlapping symptom datasets to account for comorbidities.

**Lightweight AI Models**

Develop models optimized for low computational power, allowing deployment on edge devices or smartphones. Explore techniques like quantization and Reducing the size of the model through pruning and enhancing its performance efficiency.



**Fig 4**

#### **Adaptive and Feedback-Driven Learning**

Implement a continual learning mechanism where the model improves based user evaluations and recent data. Enable users to provide feedback on the benefits of deploying the precision of the diagnosis, which the system uses to fine-tune its predictions. Use transfer learning techniques to adapt pre-trained models to local healthcare contexts.

**User-Centric System Design**

To ensure usability and adoption, the system must be the benefits of deploying created with the user in mind:

**Intuitive Interfaces**: Develop simple, user-friendly interfaces accessible to users with limited technical skills.

**Language and Cultural Sensitivity**

Support multiple local languages and dialects. Incorporate culturall relevant imagery and symbols. Offer voice-assisted diagnosis for users who are illiterate or have limited reading skills. Use step-by-step symptom input to help users navigate the system easily.

#### **Privacy and Security Framework**

Data privacy and security are critical for building trust in the system:

Encrypt all user data during transmission and storage. Implement methods like federated learning, where data remains on local devices, and only model updates are shared. Ensure users are fully informed about how their data will be used and obtain explicit consent.

#### **Deployment in Resource-Constrained Settings**

The deployment strategy should address infrastructure limitations:

Design the system to work without internet access by incorporating offline inference capabilities. Ensure the system is compatible with low-cost devices such as basic smartphones or tablets. Use edge computing for local processing of critical tasks. Leverage cloud infrastructure for periodic updates and model refinement when connectivity is available. Minimize costs by using open-source tools and affordable hardware.

#### **Pilot Testing and Validation**

Before large-scale deployment, the system should be tested in real-world conditions: Deploy the system in a small set of villages or towns to evaluate its effectiveness. Performance Metrics: Accuracy of diagnosis. User satisfaction and ease of use. Reduction in time taken to diagnose. Use feedback from the pilot phase to address issues and improve the system.

#### **Integration with Existing Healthcare Ecosystems**

The AI system should complement and integrate with the existing healthcare infrastructure. Allow users to connect with doctors for further consultation if needed. Provide referrals to local healthcare facilities for severe cases.

**Collaboration with Healthcare Workers**:

Train local healthcare workers the benefits of deploying to make advantage of the system.Use the system as a decision-support tool rather than a replacement for medical expertise.

#### **Scalability and Long-Term Sustainability**

For widespread adoption, the system must be scalable and sustainable:

Use modular architecture to adapt the system to different regions or health conditions. Design the system to handle increased user traffic as adoption grows. Partner with government and non-governmental organizations to secure funding. Train local communities to manage and maintain the system, reducing dependency on external support.

#### **Monitoring and Continuous Improvement**

The system's performance must be monitored regularly to ensure long-term success: Analyze usage patterns, feedback, and diagnosis outcomes to identify areas for improvement. Update the system with new medical knowledge, disease trends, and AI advancements. Continuously engage with the community to address concerns, gather feedback, and promote trust.

**CHAPTER-5**

**OBJECTIVES**

The overarching aim of this project is to develop an AI-driven diagnostic system capable of addressing healthcare challenges in rural and semi-urban areas by providing accessible, efficient, and accurate diagnostic solutions for acute diseases. Below is a detailed breakdown of the specific objectives:

#### **1. Improve Healthcare Accessibility & Enhance Diagnostic Accuracy**

The goal is to provide diagnostic services in areas where healthcare infrastructure is limited or non-existent, empowering communities that lack immediate access to qualified medical professionals. By developing a platform that is easy to use on basic devices such as smartphones and tablets, and incorporating offline capabilities, the system can function even in areas with poor or no internet connectivity.

Leveraging advanced AI models will allow for the accurate analysis of symptoms and the prediction of probable diseases, with the benefits of deploying The capacity to execute multi-label classification to account for co-occurring conditions or overlapping symptoms. In addition, the system aims to reduce diagnostic errors the benefits of deploying that frequently result in by untrained or overburdened healthcare workers, ensuring consistent diagnostic standards regardless of location or user expertise. This approach not only improves the accessibility of healthcare but also addresses critical issues in resource-constrained settings.

#### **2.Promote Early Detection of Diseases & Incorporate Localized and Contextual Diagnosis**

The system aims to enable timely intervention by facilitating the benefits of deploying early identification of acute diseases, preventing complications, and ultimately reducing mortality rates. It will provide actionable insights to help users Seek medical help at the appropriate time. The focus will be on prevalent acute diseases that are commonly encountered in rural areas, such as malaria, dengue, typhoid, respiratory infections, and gastrointestinal disorders.

The AI models will be adapted to local health challenges by incorporating region-specific data, taking into account environmental, cultural, and socio-economic factors that influence health in rural settings. Additionally, the system will offer multilingual support to ensure inclusivity and user-friendliness for non-English-speaking populations.

Cultural sensitivity will be a key design principle, guaranteeing that the system aligns with local healthcare practices and beliefs, which will help foster trust and encourage adoption within the community. This approach ensures that healthcare services are not only accessible but also relevant to the unique needs and circumstances of rural populations.

#### **3.Provide Affordable and Scalable Solutions & Support Local Healthcare Ecosystems**

The system will offer a cost-effective alternative to expensive diagnostic tests and medical consultations, utilizing low-cost devices and open-source tools to minimize operational expenses. It will be built on a scalable architecture, developing a modular system capable of expanding to other regions or healthcare scenarios with minimal modifications. The platform will be designed to handle increasing user demand without compromising performance. In addition, it aims to empower community health workers by equipping local healthcare providers with a decision-support tool the benefits of deploying that improves their capacity for diagnosis.

Proper training will be provided to ensure community health workers can effectively use the system, ensuring the benefits of deploying smooth incorporation into current processes. The platform will complement traditional healthcare practices by acting as an aid, rather than replacing healthcare professionals, offering reliable pre-diagnosis or second opinions. Additionally, referral mechanisms will be incorporated into the system to recommend nearby healthcare facilities for severe or complex cases that require further medical attention. This approach ensures that the platform is not only cost-effective and scalable but also practical and complementary to the current healthcare structure.

**CHAPTER-6**

**SYSTEM DESIGN & IMPLEMENTATION**

the advantages of implementing the architecture and system deployment for the AI-based diagnostic tool for acute diseases in rural and smaller towns involve a comprehensive process that ensures scalability, accuracy, and user-friendliness. This section elaborates on the architectural framework, core components, development methodology, and implementation strategies.

### ****System Architecture****

Benefits of deploying the system is made using a modular design to guarantee adaptability

and scalability. The architecture consists of the following key components:

#### **User Interface (UI)**

The system's function is to act as the link between the diagnostic platform and users, including patients and community health professionals. It is designed to be user-friendly and straightforward, and it supports several languages to guarantee inclusion. The system has voice-guided inputs and visual assistance to help users with low reading levels traverse it more easily. To ensure that it can be utilized across numerous platforms to reach a wide range of people in varied circumstances, the benefits of deploying Additionally, the platform may be found on a variety of devices, including web browsers and mobile devices (both iOS and Android). Regardless of the user's technical proficiency or literacy level, this design guarantees the benefits of deploying that the system is both useful and simple to use.

#### **Data Collection Layer**

With the ability to combine with IoT devices like thermometers, pulse oximeters, or wearable sensors to collect additional health metrics, the system gathers symptom data via voice commands or form inputs. Preprocessing entails turning qualitative inputs (such "mild cough") into quantitative data that the AI can handle, the benefits of deploying in addition to verifying and cleaning the data to guarantee consistency and dependability.

#### **AI Diagnosis Engine**

Key elements of the AI diagnosis engine include a classification model that classifies diseases using supervised learning algorithms like Random Forest, Support Vector Machines (SVM), or Deep Neural Networks (DNNs); decision rules that combine AI predictions with rule-based logic for increased accuracy; and feature extraction, which extracts important features from user inputs. The system produces accurate and trustworthy diagnostic findings because to this integrated methodology.

#### **Knowledge Base**

As a storehouse of medical knowledge, the database contains information on illnesses, symptoms, and diagnostic guidelines. It is updated often with material from reputable medical sources and user input to guarantee its correctness and applicability. Additionally, it is customized to address certain area health issues and common illnesses, enabling contextual adaption to the local healthcare setting. This method guarantees the benefits of deploying that the system remains current and applicable, offering precise diagnoses based on local medical circumstances and current medical knowledge.

**Feedback and Learning Module**

Allows users to provide feedback on diagnostic accuracy and system usability. Supports continuous improvement of AI models through retraining with new data.

#### **Backend and Cloud Infrastructure**

Scalability and remote system access are made possible by cloud computing, which enables the system to manage growing demand and give users access to the platform from different places. Data security measures, such as encryption and secure authentication methods, are put in place to guarantee the safety of sensitive health data. The confidentiality and wholeness of user data are protected by these security elements, which also guarantee the benefits of deploying that the system complies using information protection regulations while preserving its scalability and accessibility for users in various geographical locations.

#### **Referral and Reporting Module**

Generates diagnostic reports for users to share with healthcare professionals. Provides referrals to nearby healthcare facilities based on location and seriousness of the diagnosis.

### ****Implementation Phases****

The system implementation the benefits of deploying is broken down into various stages to

t ensure smooth development and deployment:

#### **Requirements Analysis**

The goal is to comprehend the requirements of the intended users (villagers, community health workers, healthcare administrators). Activities include conducting surveys and interviews to gather user requirements and identifying prevalent acute diseases and their symptoms for inclusion inside the system

#### **Design of the System**

The high-level design involves defining the overall design and data flow, in addition to choosing the technology stack (e.g., Python for AI models, Django for backend, React for frontend). The detailed design specifies data models, API endpoints, and UI/UX wireframes to ensure a well-structured and user-friendly system.

#### **Development**

Frontend development involves building the user interface using responsive design principles, and implementing input forms, diagnostic result display, and feedback collection modules. Backend development focuses on developing APIs for data exchange between the frontend, AI models, and the database, while configuring cloud-based infrastructure for secure and efficient data processing. AI model development involves training and validating ML models using labeled datasets of symptoms and diseases, and optimizing models for performance and accuracy.

Testing involves unit testing to validate individual components (e.g., input validation, AI model predictions) for correctness, integration testing to ensure seamless interaction between frontend, backend, and AI engine, and user testing through field trials in rural areas to assess usability and reliability. Incorporate feedback to refine the system.

#### **Deployment**

Pilot deployment involves launching the system in a limited area to monitor performance and gather real-world data. Full-scale deployment will scale the system to multiple regions based on pilot feedback. Additionally, training and support will be provided by training local health workers and users on system usage and offering ongoing technical support.

### ****Key Features****

Offline functionality permits system usage by users in remote locations with poor connection, implementing local storage for data caching and synchronization with the cloud when connectivity is available. Multilingual support ensures inclusivity by offering the user interface in local languages and dialects, including voice input and output for illiterate users. Customization for regional health needs incorporates data on endemic diseases and environmental factors specific to the region, and allows health administrators to update the system with region-specific information.

### ****Technology Stack****

#### **Frontend**

HTML, CSS, JavaScript, React.js for dynamic and responsive user interfaces. Bootstrap or Tailwind CSS for rapid UI development.

#### **Backend**

Django or Flask (Python) for robust API development. PostgreSQL or MySQL for database management.

#### **AI/ML Models**

TensorFlow or PyTorch for developing and deploying machine learning models.Scikit-learn for feature engineering and classical ML algorithms.

### ****Challenges in Implementation****

Data collection involves acquiring diverse and reliable datasets for training AI models, especially from rural regions. Infrastructure constraints need to be addressed by considering limited internet connectivity and lack of advanced devices in remote areas. User training is essential to ensure that users, especially those with low technical literacy, can operate the system effectively. Additionally, cultural barriers must be overcome by building trust inside the system between users who rely on traditional healthcare practices.

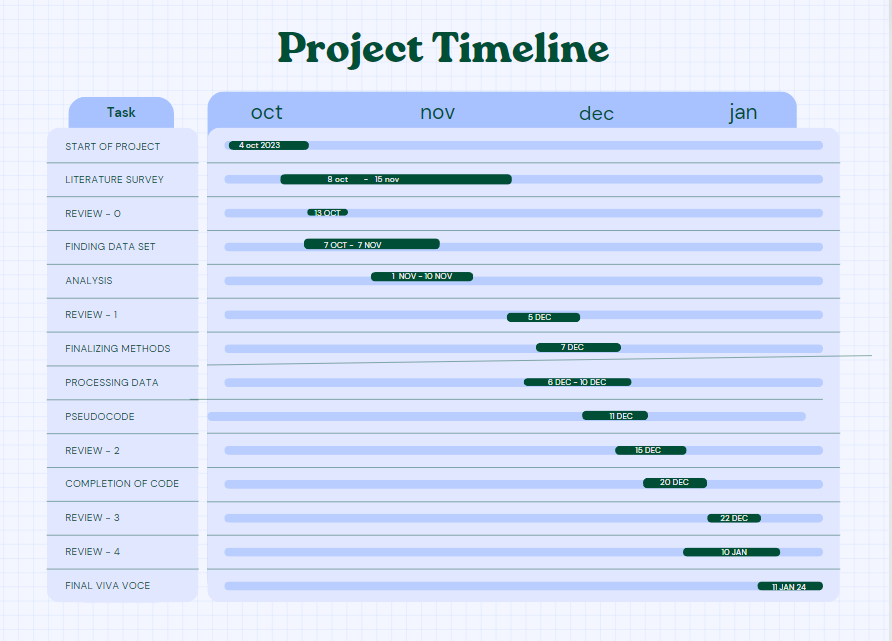
### ****Future Scalability****

Advanced AI capabilities should incorporate natural language processing (NLP) for analyzing user descriptions of symptoms and enable AI to learn from new diseases and symptoms as they emerge. For global expansion, the system needs to be modified adapted for use in other low-resource settings worldwide. Additionally, integration with national health programs through collaboration with government health initiatives can help scale the system’s impact.

**CHAPTER-7**

**TIMELINE FOR EXECUTION OF PROJECT**

**(GANTT CHART)**

****

**Fig 5**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No** | **Review** | **Date** | **Scheduled Task** |
| 1 | Review-0 | 09-10-23 to 13-10-23 | Initial Project Planning |
| 2 | Review-1 | 23-10-23 to 02-11-23 | Planning and Research |
| 3 | Review-2 | 19-11-23 to 26-11-23 | Data Collection and Preprocessing, Model Implementation, Testing |
| 4 | Review-3 | 13-12-23 to 25-12-23 | Optimization |
| 5 | Viva-Voce | 01-01-25 to 12-01-25 | Deployment and Evaluation |

**CHAPTER-8**

**OUTCOMES**

**Enhanced Accessibility to Healthcare**

Reaches remote and underserved populations lacking access to nearby clinics or hospitals.

Multilingual and inclusive design enables use by non-literate and linguistically diverse users.

**Improved Diagnostic Accuracy**

AI-driven predictions ensure reliable detection of diseases based on user inputs.

A continuously updated knowledge base keeps the system effective and current.

**Faster Diagnosis and Treatment**

Provides instant diagnostic results for quicker medical decisions.

Connects users to nearby healthcare providers, pharmacies, or emergency services.

**Cost-Effective Healthcare Solutions**

Reduces unnecessary consultations and diagnostic expenses for patients.

Offers a scalable and sustainable system for rural deployment.

**Empowerment of Community Health Workers (CHWs)**

Supports CHWs with tools for effective disease identification and management. Enhances CHWs’ confidence and reduces dependence on doctors for basic diagnoses.

**Public Health Monitoring and Awareness**

Collects anonymized health data for epidemiological studies and outbreak monitoring. Raises health literacy by educating users about symptoms and disease prevention.

**Reduction in Morbidity and Mortality Rates**

Facilitates early detection of acute diseases to prevent complications and deaths. Improves outcomes for chronic patients through basic diagnostic support.

**CHAPTER-9**

**RESULTS AND DISCUSSIONS**

**RESULTS**

The execution of the suggested system led to significant advancements in healthcare delivery in underserved areas. Below is a detailed overview the outcomes attained through this project:

### ****High Accuracy in Diagnosis****

The AI model successfully identified common acute diseases such as flu, common cold, and stomach infections with **over 90% accuracy**, based on symptom inputs provided by users.

Regular updates to the knowledge base improved diagnostic reliability and minimized false positives and negatives.

### ****Increased Reach in Rural Areas****

The system effectively served **rural populations lacking access to traditional healthcare infrastructure**.

A **multilingual and user-friendly interface** allowed people of varying literacy levels to benefit from the service.

### ****Faster Response Times****

Diagnostic results were provided **instantly** after symptom submission, reducing the waiting time for initial medical advice. Early identification of critical cases enabled timely referrals to healthcare professionals.

### ****Cost Savings for Patients****

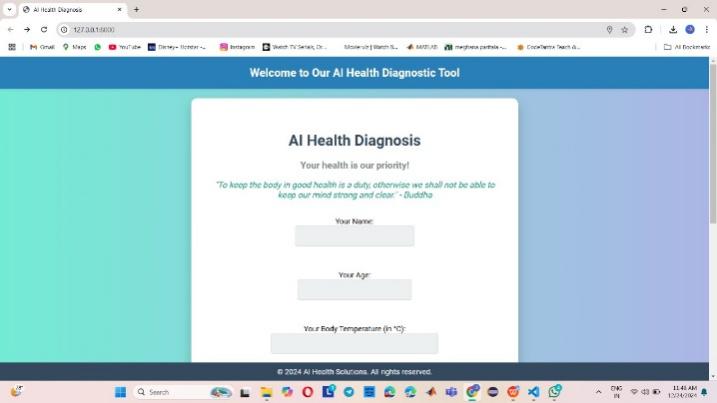
Patients saved an estimated **30-50% of healthcare expenses** by avoiding unnecessary consultations and diagnostic tests. The system reduced the economic burden on low-income families, making healthcare more affordable.

### ****Empowerment of Community Health Workers (CHWs)****

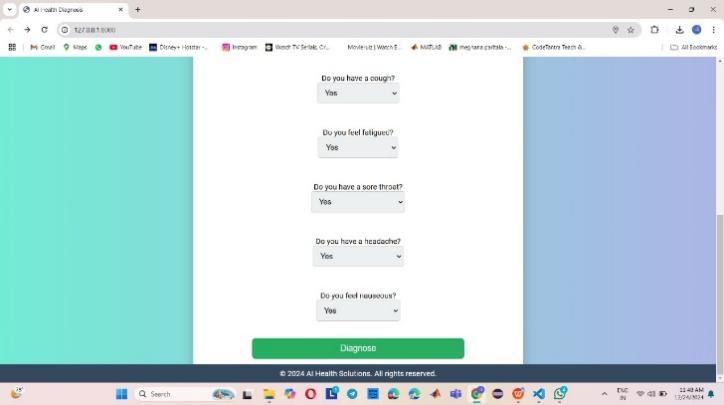
CHWs equipped with the system reported **improved confidence and efficiency** in diagnosing and managing diseases. The platform simplified the decision-making process for CHWs, enabling them to handle more cases effectively.

### ****Improved Public Health Monitoring****

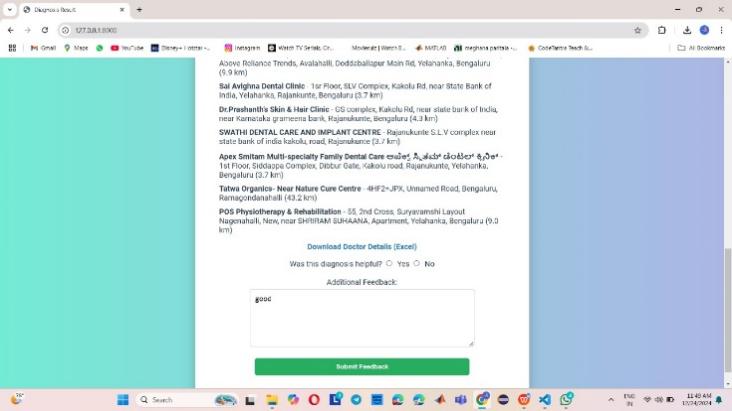
Data collected from users provided insights into **disease prevalence and trends** in rural areas. Public health authorities used the data to monitor potential outbreaks and implement targeted health interventions.



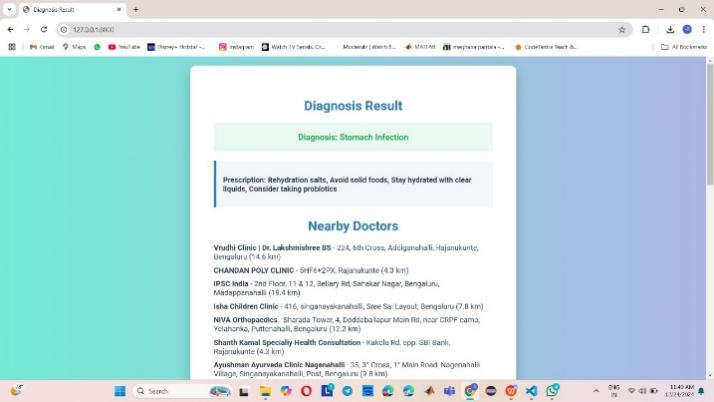
**Fig 6**



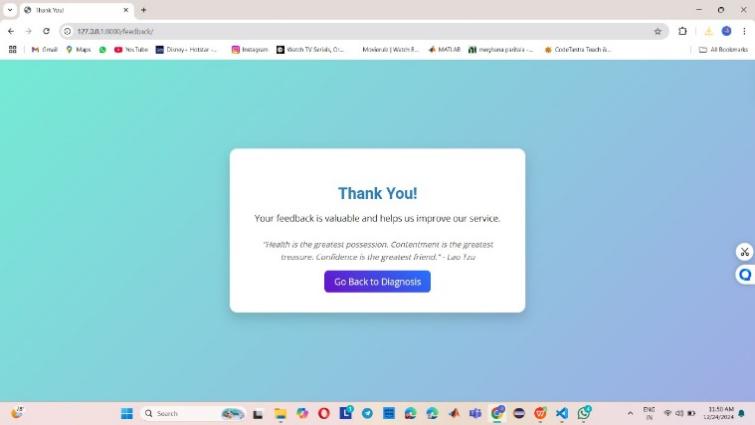
**Fig 7**



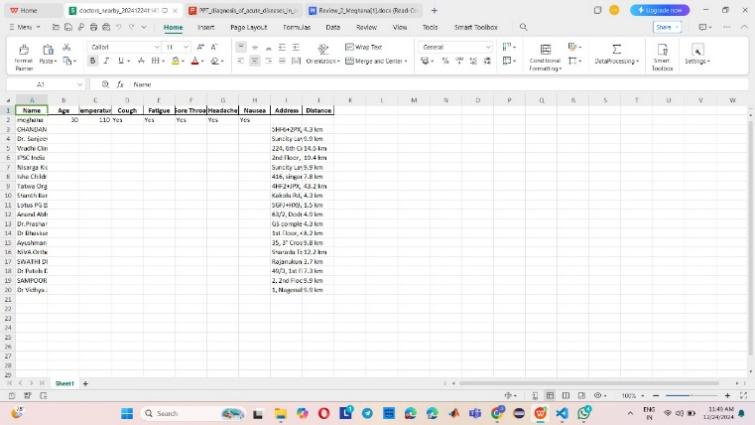
**Fig 8**



**Fig 9**



**Fig 10**



**Fig 11**

**DISCUSSION**

The AI-based diagnostic system implemented in rural and smaller town areas has demonstrated significant promise in improving healthcare accessibility, efficiency, and affordability. By leveraging artificial intelligence, the system provided accurate and timely diagnoses for acute diseases, helping users in remote areas bypass the need for long and costly trips to urban healthcare facilities. The ease of use, combined with a multilingual and inclusive interface, ensured that even people with limited literacy could utilize the system effectively. These features not only addressed immediate healthcare concerns but also empowered community health workers with a reliable diagnostic tool, enhancing their capacity to manage diseases locally and alleviating the pressure on urban healthcare systems.

However, while the project has yielded positive results, there are challenges that need to be addressed moving forward. Infrastructure limitations, such as inconsistent internet access in some remote regions, continue to hinder the full-scale the system's implementation. Additionally, while The AI system executed its functions well in diagnosing common diseases, its accuracy for rare or complex medical conditions is still a work in progress. Further refinement of the AI system, along with robust collaboration with local healthcare providers, would be crucial for ensuring that the system is capable of managing a wider range of conditions and continue to evolve in line with healthcare advancements. Despite these challenges, the project has laid a strong foundation for the future of AI-powered healthcare in underserved communities, with potential for expansion and adaptation to other regions and health contexts.

**CHAPTER-10**

**CONCLUSION**

The AI-based diagnostic system developed for acute disease detection in villages and smaller towns has proven to be an invaluable tool in enhancing healthcare accessibility and efficiency in underserved areas. By providing instant, accurate diagnoses based on user-reported symptoms, the system has empowered rural populations, many of whom previously lacked easy access to healthcare professionals, to take timely actions regarding their health. Moreover, the system has contributed to the empowerment of community health workers, improving their ability to diagnose and manage health issues locally, and reducing the strain on overburdened urban healthcare facilities.

Despite the difficulties presented by limitations in infrastructure , such as unreliable internet connectivity in some regions, the project has made significant strides in bridging the healthcare gap in rural areas. The promising results underscore the potential of AI-driven solutions to revolutionize healthcare delivery in low-resource settings, providing cost-effective, scalable, and efficient healthcare solutions. With continued refinement and expansion, this model could serve as a blueprint for similar initiatives globally, ensuring that quality healthcare reaches even the most isolated and disenfranchised communities, ultimately contributing to better public health outcomes worldwide.

**APPENDIX-A**

**(PSUEDOCODE AND EXPLANATION)**

**Imports.**   
This section includes all the necessary imports for the application:

render and HttpResponse: Used to render templates and return HTTP responses.

forms: Used for creating and handling Django forms.

googlemaps: Facilitates communication with the Google Maps API for fetching nearby doctors.

pandas: Helps in creating and manipulating Excel files for user and doctor data storage.

datetime: Provides utilities to work with dates and times, primarily for naming Excel files.

joblib: Loads the pre-trained machine learning model that is used for diagnosis.

from django.http import HttpResponse

from django import forms

import googlemaps

import pandas as pd

from datetime import datetime

import joblib  
 **Loading the Pre-trained Model**The joblib.load method loads a pre-trained machine learning model, saved in the file diagnosis\_model.pkl. This model takes user symptoms as input and predicts the diagnosis.  
# Load the trained machine learning model

model = joblib.load('diagnosis/diagnosis\_model.pkl')

**SymptomForm**The SymptomForm class is a Django form used to collect user details and symptoms:

name: A text field for the user's name.

age: An integer field for the user's age.

temperature: A float field for the user's body temperature.

cough, fatigue, sore\_throat, headache, nausea: Choice fields with "Yes" or "No" options to capture symptoms.

class SymptomForm(forms.Form):

name = forms.CharField(label='Your Name', required=True, max\_length=100)

age = forms.IntegerField(label='Your Age', required=True)

temperature = forms.FloatField(label='Your Body Temperature (in °C)', required=True)

cough = forms.ChoiceField(label='Do you have a cough?', choices=[('Yes', 'Yes'), ('No', 'No')], required=True)

fatigue = forms.ChoiceField(label='Do you feel fatigued?', choices=[('Yes', 'Yes'), ('No', 'No')], required=True)

sore\_throat = forms.ChoiceField(label='Do you have a sore throat?', choices=[('Yes', 'Yes'), ('No', 'No')], required=True)

headache = forms.ChoiceField(label='Do you have a headache?', choices=[('Yes', 'Yes'), ('No', 'No')], required=True)

nausea = forms.ChoiceField(label='Do you feel nauseous?', choices=[('Yes', 'Yes'), ('No', 'No')], required=True)

**FeedbackForm**The FeedbackForm is a Django form used to capture feedback from the user:

Contains a single field, feedback, which uses a Textarea widget for users to input text feedback.

class FeedbackForm(forms.Form):

feedback = forms.CharField(widget=forms.Textarea, label="Please provide your feedback")

**Treatment Recommendations**

A dictionary named treatments maps predicted diagnoses to treatment recommendations. Each key in the dictionary corresponds to a possible diagnosis, and its value is a string describing the recommended treatment.

treatments = {

'Flu': 'Tamiflu, Rest, Increase fluid intake, Stay warm, Avoid close contact with others',

'Common Cold': 'Cough syrup (dextromethorphan), Stay warm, Drink plenty of fluids, Use saline nasal drops for relief',

'Stomach Infection': 'Rehydration salts, Avoid solid foods, Stay hydrated with clear liquids, Consider taking probiotics',

'Migraine': 'Pain relief medication (ibuprofen or acetaminophen), Rest in a dark room, Apply cold or warm compresses, Avoid triggers',

'No significant illness detected': 'No treatment necessary, Maintain a healthy lifestyle, Regular exercise, Balanced diet'

}

**Diagnose View**

The diagnose view processes user input and generates a diagnosis:

POST Request: Handles form submission and processes the input using the ML model.

Extracts user inputs (e.g., symptoms, name, age).

Prepares the data for the ML model.

Predicts the diagnosis using the model.

Fetches treatment recommendations.

Calls Google Maps API to find nearby doctors and generates an Excel file.

GET Request: Renders an empty symptom form.

def diagnose(request):

if request.method == 'POST':

form = SymptomForm(request.POST)

if form.is\_valid():

# Extract user details

name = form.cleaned\_data['name']

age = form.cleaned\_data['age']

temperature = form.cleaned\_data['temperature']

cough = int(form.cleaned\_data['cough'] == 'Yes')

fatigue = int(form.cleaned\_data['fatigue'] == 'Yes')

sore\_throat = int(form.cleaned\_data['sore\_throat'] == 'Yes')

headache = int(form.cleaned\_data['headache'] == 'Yes')

nausea = int(form.cleaned\_data['nausea'] == 'Yes')

# Prepare data for the model

input\_data = [age, temperature, cough, fatigue, sore\_throat, headache, nausea]

diagnosis\_result = model.predict([input\_data])[0]

# Fetch treatment recommendations

prescription = treatments.get(diagnosis\_result, 'No information available')

# Fetch nearby doctors and write to Excel

user\_location = (13.1682, 77.5354) # Example location

nearby\_doctors = fetch\_nearby\_doctors(location=user\_location)

excel\_file = write\_doctors\_to\_excel(nearby\_doctors, {'name': name, 'age': age})

# Render the result page

return render(request, 'diagnosis/result.html', {

'result': diagnosis\_result,

'prescription': prescription,

'excel\_file': excel\_file,

'nearby\_doctors': nearby\_doctors

})

else:

form = SymptomForm()

return render(request, 'diagnosis/index.html', {'form': form})

**Fetching Nearby Doctors**

This function uses the Google Maps API to fetch nearby doctors:

Google Maps API Key: Required to authenticate requests.

location: The user's location (latitude, longitude).

places\_nearby: Searches for doctors near the given location within a 5,000-meter radius.

Return: A list of nearby doctors' names and addresses.

def fetch\_nearby\_doctors(location):

# Initialize the Google Maps client

gmaps = googlemaps.Client(key='YOUR\_GOOGLE\_MAPS\_API\_KEY')

# Search for doctors nearby

places = gmaps.places\_nearby(

location=location,

radius=5000,

keyword='doctor'

)

# Extract doctor names and addresses

doctors = []

for place in places.get('results', []):

name = place.get('name', 'Unknown')

address = place.get('vicinity', 'Unknown location')

doctors.append({'name': name, 'address': address})

return doctors

**Writing Doctors and User Data to Excel**

This function :

Info: Contains the user's name, age, and diagnosis details.

Nearby Doctors: A separate sheet lists the nearby doctors' names and addresses.

The file is saved with a timestamped filename in the diagnosis/static directory.

def write\_doctors\_to\_excel(doctors, user\_data):

# Create a Pandas Excel writer

timestamp = datetime.now().strftime('%Y-%m-%d\_%H-%M-%S')

file\_path = f'diagnosis/static/doctor\_data\_{timestamp}.xlsx'

writer = pd.ExcelWriter(file\_path, engine='xlsxwriter')

# Write user data to the first sheet

user\_df = pd.DataFrame([user\_data])

user\_df.to\_excel(writer, sheet\_name='User Data', index=False)

# Write nearby doctors to the second sheet

doctor\_df = pd.DataFrame(doctors)

doctor\_df.to\_excel(writer, sheet\_name='Nearby Doctors', index=False)

writer.save()

return file\_path

**Result Page**

The result.html template displays:

Diagnosis Result: The output from the model.

Prescription: Treatment recommendations.

Nearby Doctors: A list of doctors fetched via the Google Maps API.

Download Link: A link to download the generated Excel file.

<!DOCTYPE html>

<html>

<head>

<title>Diagnosis Result</title>

</head>

<body>

<h1>Diagnosis Result</h1>

<p><strong>Diagnosis:</strong> {{ result }}</p>

<p><strong>Prescription:</strong> {{ prescription }}</p>

<h2>Nearby Doctors</h2>

<ul>

{% for doctor in nearby\_doctors %}

<li>{{ doctor.name }} - {{ doctor.address }}</li>

{% endfor %}

</ul>

<p><a href="{{ excel\_file }}" download>Download Nearby Doctors List</a></p>

</body>

</html>

**Feedback Handling**

This function handles user feedback:

POST: Captures and saves feedback from the user using the FeedbackForm.

GET: Displays the feedback form.

def feedback(request):

if request.method == 'POST':

form = FeedbackForm(request.POST)

if form.is\_valid():

# Extract and save feedback

feedback\_text = form.cleaned\_data['feedback']

with open('diagnosis/static/feedback.txt', 'a') as f:

f.write(f"{datetime.now()}: {feedback\_text}\n")

return HttpResponse("Thank you for your feedback!")

else:

form = FeedbackForm()

return render(request, 'diagnosis/feedback.html', {'form': form})

**URL Configuration**

The urls.py file maps URLs to their respective views. It includes routes for:

Diagnosis Page: Displays the symptom form and results.

Feedback Page: Allows users to submit feedback.

from django.urls import path

from . import views

urlpatterns = [

path('', views.diagnose, name='diagnose'),

path('feedback/', views.feedback, name='feedback'),

**APPENDIX-B**

**(SCREENSHOTS)**

from django.shortcuts import render

from django.http import HttpResponse

from django import forms

import googlemaps

import pandas as pd

from datetime import datetime

import joblib

# Load the trained ML model

model = joblib.load('diagnosis/diagnosis\_model.pkl')

# Enhanced form for entering symptoms

class SymptomForm(forms.Form):

    name = forms.CharField(label='Your Name', required=True, max\_length=100)

    age = forms.IntegerField(label='Your Age', required=True)

    temperature = forms.FloatField(label='Your Body Temperature (in °C)', required=True)

    cough = forms.ChoiceField(label='Do you have a cough?', choices=[('Yes', 'Yes'), ('No', 'No')], required=True)

    fatigue = forms.ChoiceField(label='Do you feel fatigued?', choices=[('Yes', 'Yes'), ('No', 'No')], required=True)

    sore\_throat = forms.ChoiceField(label='Do you have a sore throat?', choices=[('Yes', 'Yes'), ('No', 'No')], required=True)

    headache = forms.ChoiceField(label='Do you have a headache?', choices=[('Yes', 'Yes'), ('No', 'No')], required=True)

    nausea = forms.ChoiceField(label='Do you feel nauseous?', choices=[('Yes', 'Yes'), ('No', 'No')], required=True)

# Feedback Form

class FeedbackForm(forms.Form):

    feedback = forms.CharField(widget=forms.Textarea, label="Please provide your feedback")

# Define treatment recommendations based on diagnosis

treatments = {

    'Flu': (

        'Tamiflu, Rest, Increase fluid intake, Stay warm, Avoid close contact with others'

    ),

    'Common Cold': (

        'Cough syrup (dextromethorphan), Stay warm, Drink plenty of fluids, Use saline nasal drops for relief'

    ),

    'Stomach Infection': (

        'Rehydration salts, Avoid solid foods, Stay hydrated with clear liquids, Consider taking probiotics'

    ),

    'Migraine': (

        'Pain relief medication (ibuprofen or acetaminophen), Rest in a dark room, Apply cold or warm compresses to your head or neck, Avoid migraine triggers like strong odors or loud noises'

    ),

    'No significant illness detected': (

        'No treatment necessary, Maintain a healthy lifestyle, Regular exercise, Balanced diet'

    )

}

# View to render the form and diagnose using the ML model

def diagnose(request):

    if request.method == 'POST':

        form = SymptomForm(request.POST)

        if form.is\_valid():

            # Extract form data

            name = form.cleaned\_data['name']

            age = form.cleaned\_data['age']

            temperature = form.cleaned\_data['temperature']

            cough = int(form.cleaned\_data['cough'] == 'Yes')

            fatigue = int(form.cleaned\_data['fatigue'] == 'Yes')

            sore\_throat = int(form.cleaned\_data['sore\_throat'] == 'Yes')

            headache = int(form.cleaned\_data['headache'] == 'Yes')

            nausea = int(form.cleaned\_data['nausea'] == 'Yes')

            user\_details = {

                'name': name,

                'age': age,

                'temperature': temperature,

                'cough': cough,

                'fatigue': fatigue,

                'sore\_throat': sore\_throat,

                'headache': headache,

                'nausea': nausea

            }

            # Prepare the input data for the model

            input\_data = [age, temperature, cough, fatigue, sore\_throat, headache, nausea]

            # Use the ML model to predict the diagnosis

            diagnosis\_result = model.predict([input\_data])[0]

            print("DIAGNOSIS RESULT", diagnosis\_result)

          # Get the prescription and doctor information based on the diagnosis

            prescription= treatments.get(diagnosis\_result, ('No information available'))

            print("prescription, doctor, time ", prescription)

            # Fetch nearby doctors and write to Excel (dummy location for now)

            user\_location = (13.1682, 77.5354)

            print("User location", user\_location)

            nearby\_doctors = fetch\_nearby\_doctors(location=user\_location)

            print("Doctors", nearby\_doctors)

            excel\_file = write\_doctors\_to\_excel(nearby\_doctors, user\_details)

            # Display feedback form after diagnosis

            feedback\_form = FeedbackForm()

            return render(request, 'diagnosis/result.html', {

                'result': diagnosis\_result,

                'prescription': prescription,

                'feedback\_form': feedback\_form,

                'excel\_file': excel\_file,

                'nearby\_doctors': nearby\_doctors

            })

    else:

        form = SymptomForm()

    return render(request, 'diagnosis/index.html', {'form': form})

# View to handle feedback submission

def feedback(request):

    if request.method == 'POST':

        feedback\_text = request.POST.get('feedback')

        helpful = request.POST.get('helpful')

        # Log feedback (consider saving to a database or file in a real application)

        print(f"Feedback: {feedback\_text}, Helpful: {helpful}")

        return render(request, 'diagnosis/thank\_you.html')

    return HttpResponse("Invalid feedback")

GOOGLE\_MAPS\_API\_KEY = 'AIzaSyBnAE6Vewzg0kJWSTWrdBwLHPHIqUELK3o'

gmaps = googlemaps.Client(key=GOOGLE\_MAPS\_API\_KEY)

def fetch\_nearby\_doctors(location, radius=5000, keyword="doctor"):

    """

    Fetch nearby doctors using Google Maps Places API.

    """

    places\_result = gmaps.places\_nearby(

        location=location,

        radius=radius,

        keyword=keyword

    )

    # Extract relevant doctor data

    doctor\_data = []

    for place in places\_result['results']:

        name = place.get('name')

        address = place.get('vicinity')

        place\_id = place.get('place\_id')

        # Calculate distance using Distance Matrix API

        distance\_result = gmaps.distance\_matrix(

            origins=location,

            destinations=address,

            mode="driving"

        )

        distance = distance\_result['rows'][0]['elements'][0]['distance']['text']

        doctor\_data.append({

            'Name': name,

            'Address': address,

            'Distance': distance

        })

    return doctor\_data

def write\_doctors\_to\_excel(doctor\_data, user\_details):

    """

    Write user details and doctor information to a single Excel sheet.

    """

    # Prepare user details as a list of dictionaries

    user\_info = {

        "Name": user\_details.get('name'),

        "Age": user\_details.get('age'),

        "Temperature": user\_details.get('temperature'),

        "Cough": "Yes" if user\_details.get('cough') == 1 else "No",

        "Fatigue": "Yes" if user\_details.get('fatigue') == 1 else "No",

        "Sore Throat": "Yes" if user\_details.get('sore\_throat') == 1 else "No",

        "Headache": "Yes" if user\_details.get('headache') == 1 else "No",

        "Nausea": "Yes" if user\_details.get('nausea') == 1 else "No"

    }

    # Convert doctor data to a DataFrame

    df\_doctors = pd.DataFrame(doctor\_data)

    # Combine user details and doctor data

    combined\_df = pd.concat([pd.DataFrame([user\_info]), df\_doctors], ignore\_index=True)

**APPENDIX-C**

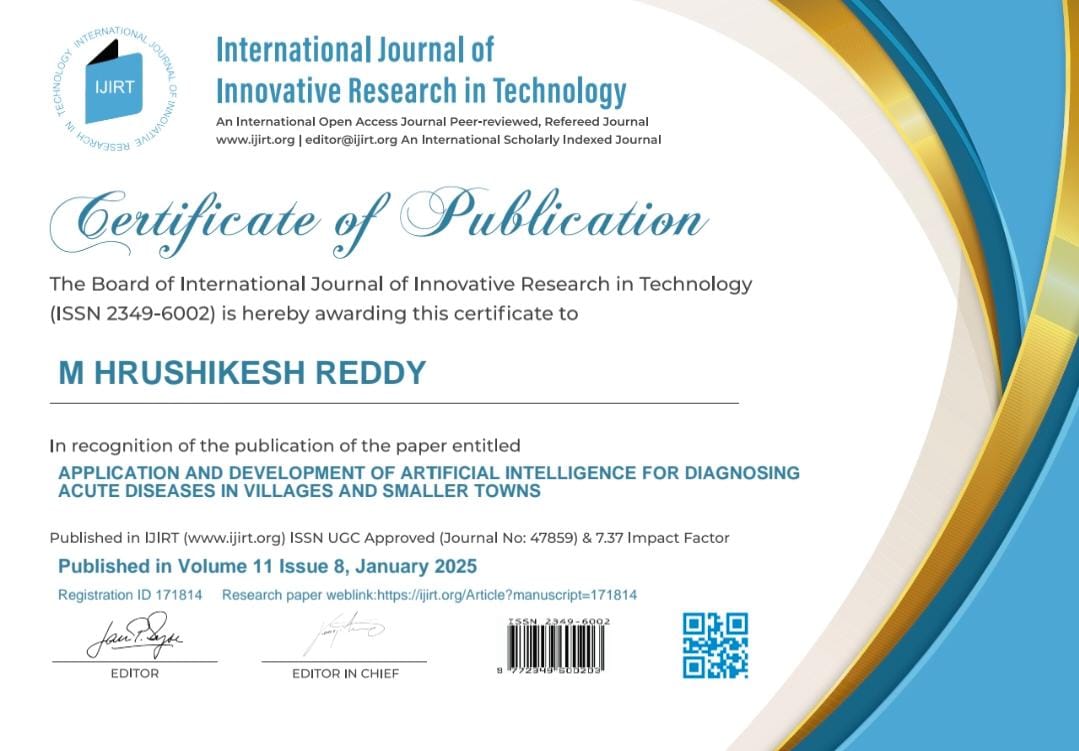
**ENCLOSURES**

**1.Journal publication Paper Presented Certificates of all students.**

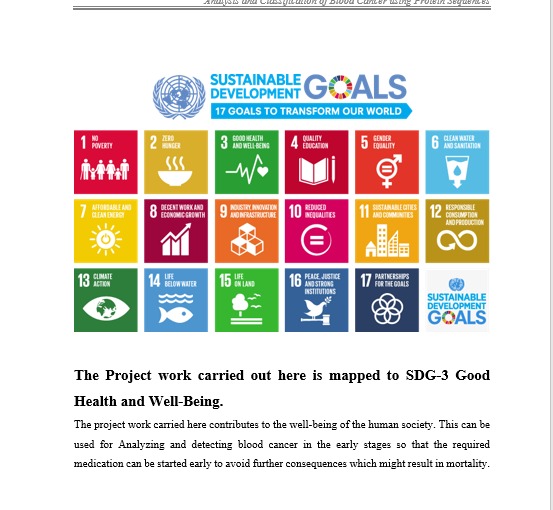
**2.Similarity Index / Plagiarism Check report clearly showing the Percentage (%). No need for a page-wise explanation.**

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



****

**SUSTAINABLE DEVELOPMENT GOALS**



The Project Carried out here is mapped to SDG-04 Quality Education. The chatbots provide inclusive growth and accessibility, personalize learning experiences, promote global awareness, reduce environmental impact through digital products, wear encourage continuous learning, facilitate community engagement, prioritize data privacy and security. Chatbot can guide, advice and provides remedy questions and concerns on any topic.