KISHKINDA UNIVERSITY, BALLARI

Established under the Karnataka Act No. 20 of 2023 | Recognised by Govt. of Karnataka & UGC, New Delhi

T.E.H.R.D TRUST

KISHKINDA UNIVERSITY
FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MTech-Third Semester

Mini Project Report on

" MEDICAL DIAGNOSIS CHATBOT USING RECURSIVE NEURAL NETWORK (RNN) ARCHITECTURE "

Submitted by,

Mr. S MD Zaheed Hussain
KUB24MSC018

Under the Guidance of,

Dr. Yerriswamy T
Professor
Department of Computer Science and Engineering
Kishkinda University

Main Campus
Mount View Campus, Off 28Kms, Ballari - Siruguppa Road,
Near Sindhigeri, GP NO.735, Hagaluru, Siruguppa Taluk,
Ballari - 583120, Karnataka

KISHKINDA UNIVERSITY

Advancing Knowledge Transforming Lives

KISHKINDA UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY(FET)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini-Project entitled "**MEDICAL DIAGNOSIS CHATBOT USING RECURSIVE NEURAL NETWORK (RNN) ARCHITECTURE**" has been successfully presented by **Mr. S MD Zaheed Hussain** bearing USN **KUB24MCS018** a student of III semester M.Tech for the partial fulfillment of the requirements for the **Master Degree in Computer Science & Engineering** of the KISHKINDA UNIVERSITY during the academic year 2024-2025.

Dr. Yerriswamy T	**Dr. Rajashree V Biradar**
Guide	**Chairperson**
Dept of Computer Science and Engineering	**Dept of Computer Science and Engineering**
Kishkinda University	**Kishkinda University**

ACKNOWLEDGEMENT

It is my privilege and primary duty to express my heartfelt gratitude and respect to all those who guided and inspired me in the successful completion of this Mini Project.

I express my sincere thanks to **Dr. Yerriswamy T, Under who's guidance I completed this project** and also initiated it. And also express my sincere thanks to **Dr. Rajashree V. Biradar, my Guide and Chairperson of the Department of Computer Science, Kishkinda University, Ballari**, for her valuable guidance, constant support, and encouragement throughout the course of this project. I am also grateful to **Dr. V C Patil, Dean, Faculty of Engineering and Technology (FET), Kishkinda University**, for his continuous support and motivation. My sincere thanks to **Dr. Eranna**, **Registrar & Pro Vice Chancellor, Kishkinda University**, for extending all necessary administrative support during the project work. I would like to extend my gratitude to **Dr. T N Nagabhushana, Hon'ble Vice-Chancellor, Kishkinda University, Ballari**, for providing a stimulating academic environment and encouragement. I would like to extend my gratitude to **Mr. Y J Prithviraj Bhupal, Hon'ble Pro-Chancellor, Kishkinda University, Ballari**, for providing a stimulating academic environment and encouragement.

I am deeply thankful to the **Management of Kishkinda University, Ballari**, for providing the necessary infrastructure and facilities to carry out this Mini Project.

I also wish to express my appreciation to the **teaching and non-teaching staff of the Department of Computer Science and Engineering** for their cooperation and assistance.

Lastly, I extend my heartfelt gratitude to all those who have directly or indirectly contributed their efforts in making this Mini Project a success.

Mr. S MD Zaheed Hussain
KUB24MCS018

ABSTRACT

India’s road infrastructure faces persistent challenges due to potholes, cracks, and surface deterioration, which compromise commuter safety, increase vehicle maintenance costs, and strain municipal budgets. Traditional inspection methods, such as manual surveys and specialized monitoring vehicles, are slow, labor-intensive, and unsuitable for large-scale deployment. To address this, we propose RoadCare+, a lightweight and efficient road surface damage detection framework tailored for Indian roads. The system leverages an enhanced YOLOv7 architecture—referred to as GRC-YOLOv7—which integrates three architectural improvements: GhostNet, BasicRFB (Receptive Field Block), and CARAFE (Content-Aware ReAssembly of Features). GhostNet reduces computational redundancy, BasicRFB enriches contextual feature extraction for small or thin cracks, and CARAFE improves upsampling quality for multi-scale feature reconstruction. Together, these modules deliver a balance of high detection accuracy, reduced model size, and real-time inference speed, making the solution deployable on both GPUs and resource-constrained edge devices.

The framework was trained exclusively on the RDD2020 India dataset, which contains annotated images of Indian road conditions. Data preprocessing techniques—including resizing, augmentation, and YOLO-format annotation conversion—were applied to enhance robustness across varying lighting and environmental conditions. The trained model was evaluated using metrics such as mAP@0.5, precision, recall, F1-score, FPS, and parameter count. Experimental results show that GRC-YOLOv7 achieves higher detection accuracy and faster inference speeds compared to the baseline YOLOv7 model, while maintaining a significantly lower computational load. Beyond detection, RoadCare+ includes an automated reporting pipeline that generates Excel and PDF summaries of detected damages with bounding box details, severity scores, and confidence values. An optional Streamlit-based web interface further enhances accessibility, allowing municipal engineers and civic authorities to upload road images and directly obtain actionable reports without requiring technical expertise.

The proposed solution demonstrates a practical and scalable approach to road damage monitoring in the context of smart city infrastructure management. By enabling earlier detection and prioritization of repairs, RoadCare+ can reduce inspection costs, improve commuter safety, and support data-driven urban planning. Future work includes extending the framework to multi-class road anomalies, real-time IoT integration with vehicular dashcams, and large-scale deployment in city-wide transportation networks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The rapid advancement of artificial intelligence and natural language processing has significantly changed the way humans interact with computer systems. In the healthcare domain, these technologies have enabled the development of intelligent applications that can assist users in understanding health conditions, identifying symptoms, and obtaining preliminary medical guidance. With the growing demand for quick and accessible healthcare information, digital solutions such as chatbots have emerged as effective tools to bridge the gap between patients and medical knowledge.

Many individuals experience health-related symptoms but often face challenges such as limited access to medical professionals, long waiting times, high consultation costs, or uncertainty regarding the seriousness of their condition. As a result, people may delay seeking professional medical advice, which can lead to complications. An automated system capable of providing basic symptom analysis and guidance can help users make informed decisions at an early stage.

The Symptom Checker Chatbot is an artificial intelligence–based conversational system designed to interact with users through natural language. It allows users to describe their symptoms in simple text form and processes the input using natural language processing techniques and a recurrent neural network model. Based on the detected intent and symptom patterns, the system generates relevant responses and provides general health-related information.

This project aims to demonstrate the practical application of machine learning, neural networks, and NLP in building an intelligent healthcare support system. The chatbot is implemented using Python, PyTorch, NLTK, and Flask, making it lightweight, scalable, and easy to deploy as a web-based application. While the system does not replace professional medical diagnosis or treatment, it serves as a useful preliminary advisory tool that enhances accessibility to health information.

Overall, the Symptom Checker Chatbot represents an effective integration of artificial intelligence into healthcare assistance, promoting early awareness, reducing unnecessary hospital visits, and providing users with timely and reliable information in a user-friendly manner.

CHAPTER 2

OBJECTIVES

1. To design and develop an intelligent chatbot capable of understanding user-entered health symptoms using natural language processing techniques.
2. To implement a machine learning–based model (RNN) for accurate intent classification and appropriate response generation.
3. To provide users with quick and reliable preliminary health-related information through an interactive conversational interface.
4. To integrate location-based functionality for suggesting nearby medical centers to users when professional consultation is required.
5. To build a user-friendly and scalable web application using Flask that demonstrates the practical application of artificial intelligence in healthcare support systems.

CHAPTER 3

LITERATURE SURVEY

The application of artificial intelligence in healthcare has gained significant attention in recent years due to its potential to improve accessibility, efficiency, and quality of medical services. Among various AI-driven solutions, symptom checker systems and healthcare chatbots have emerged as effective tools for providing preliminary medical guidance and assisting users in understanding their health conditions.

Early symptom checker systems were primarily rule-based, relying on predefined decision trees and manually created medical rules to match symptoms with possible diseases. Although these systems provided basic functionality, they lacked flexibility and were unable to handle complex or ambiguous user inputs. Furthermore, they required extensive manual updates to incorporate new medical knowledge, making them less scalable and adaptive.

With advancements in machine learning and natural language processing, researchers began developing intelligent chatbot systems capable of understanding user queries expressed in natural language. Studies have shown that NLP-based chatbots improve user experience by allowing conversational interaction instead of rigid form-based input. Tokenization, stemming, bag-of-words models, and intent classification techniques have been widely used to process and interpret textual health queries.

Several researchers have explored the use of neural networks, particularly Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, for medical chatbot applications. These models are effective in capturing sequential dependencies in language and improving intent detection accuracy. Neural network-based chatbots have demonstrated higher adaptability and better response quality compared to traditional rule-based approaches.

Recent studies also emphasize integrating healthcare chatbots with web frameworks such as Flask or Django to enable real-time interaction through browser-based interfaces. This approach allows easy deployment and accessibility while maintaining system scalability. Some advanced systems further integrate external APIs, geolocation services, and electronic health record databases to enhance the usefulness of the chatbot by providing hospital recommendations and personalized health insights.

Despite these advancements, existing systems face limitations such as restricted medical datasets, inability to handle rare symptoms, lack of emotional intelligence, and dependency on training data quality. Moreover, most symptom checker chatbots are intended for informational purposes and cannot replace professional medical diagnosis, highlighting the importance of ethical considerations in AI-based healthcare applications.

The proposed Symptom Checker Chatbot builds upon these existing research works by employing natural language processing and a recurrent neural network model to interpret user symptoms and generate appropriate responses. By integrating a location-based medical center recommendation feature and a web-based interface, the project aims to provide a practical, user-friendly, and scalable preliminary healthcare assistance system.

CHAPTER 4

PROBLEM STATEMENT

To design and develop an intelligent, web-based symptom checker chatbot that can understand user-entered health queries using natural language processing techniques, analyze reported symptoms through a machine learning model, and provide reliable preliminary medical information along with suggestions for nearby medical centers, thereby enabling users to obtain quick and accessible healthcare guidance without replacing professional medical consultation.

CHAPTER 5

METHODOLOGY

The methodology for developing the Symptom Checker Chatbot follows an incremental sprint-based strategy, where the project is divided into multiple development phases. Each sprint focuses on a specific functional and technical component of the system, enabling continuous validation, improvement, and integration of features such as natural language understanding, model training, response generation, web deployment, and medical center recommendation. This approach ensures systematic development, modular integration, and deployment readiness.

The broad development phases are as follows:

- **Sprint 1:** Dataset collection and intent design
- **Sprint 2:** NLP preprocessing pipeline development
- **Sprint 3:** Baseline RNN model implementation and training
- **Sprint 4:** Model optimization and intent classification tuning
- **Sprint 5:** Chatbot logic and response generation module
- **Sprint 6:** Medical center recommendation system integration
- **Sprint 7:** Flask-based web application development
- **Sprint 8:** System testing, validation, and performance evaluation
- **Sprint 9:** Final integration, documentation, and report preparation

This sprint-based methodology enables parallel development of AI models, backend logic, and user interface components, transforming the chatbot into a complete healthcare assistance system rather than a standalone machine learning model.

5.1 Dataset Acquisition and Intent Engineering

The dataset forms the foundation of the chatbot’s intelligence. An intent-based dataset was created using a structured JSON format.

Dataset Overview

- **Format:** JSON (intents.json)
- **Type:** Text-based conversational dataset
- **Components per intent:**
    - Tag (intent label)

   - Input patterns (user queries)
   - Responses (bot replies)

Intent Categories Include:

- Greetings
- Symptom queries (fever, headache, cough, fatigue, etc.)
- Emergency queries
- Medical assistance
- General health information

This structure allows mapping of diverse natural language queries into well-defined categories for efficient classification.

5.2 NLP Preprocessing Pipeline

To convert human language into machine-readable form, a preprocessing pipeline was developed using NLTK.

Preprocessing Steps:

1. **Tokenization:**
   Splits sentences into individual words.
2. **Lowercasing:**
   Standardizes text to avoid case sensitivity.
3. **Stemming:**
   Converts words to root form (e.g., "running" → "run").
4. **Noise Removal:**
   Removes punctuation and special characters.
5. **Bag-of-Words Vectorization:**
   Converts tokens into fixed-length numerical vectors representing word presence.

This pipeline ensures consistency between training data and real-time user input and significantly improves intent classification accuracy.

5.3 Model Design – RNN-Based Intent Classifier

A Recurrent Neural Network (RNN) model was selected due to its ability to process sequential textual data.

Architecture:

- Input Layer – Bag-of-words vector

- Hidden Layers – RNN units
- Output Layer – Softmax classifier (intent probabilities)

Advantages:

- Captures contextual dependencies
- Lightweight and fast
- Suitable for real-time chatbot applications

5.4 Model Training and Optimization

Training Environment

- Framework: PyTorch
- Language: Python
- Libraries: NLTK, NumPy, Torch

Hyperparameters

- Batch size: 8
- Epochs: 100
- Learning rate: 0.001
- Optimizer: Adam
- Loss Function: Cross-Entropy Loss

Training Steps:

1. Load dataset
2. Apply NLP preprocessing
3. Generate feature-label pairs
4. Train RNN using backpropagation
5. Validate prediction accuracy
6. Save trained model as data_rnn.pth

Optimization Techniques:

- Learning rate tuning
- Overfitting control through dataset expansion
- Confidence thresholding during inference

5.5 Chatbot Logic and Inference Flow

The chatbot logic is implemented in chat.py.

Runtime Processing Flow:

1. Receive user message
2. Apply NLP preprocessing
3. Convert to feature vector
4. Load trained RNN model
5. Predict intent
6. Compute confidence score
7. Select response from intent dataset
8. Trigger medical center module if required
9. Send final response to UI

This logic ensures fast and accurate interaction.

5.6 Medical Center Recommendation Module

To enhance practical usability, a location-based recommendation system is integrated.

Working:

- User location detected using geocoder
- Medical centers loaded from medical_centers.json
- Nearest centers filtered
- Details appended to chatbot response

This bridges the gap between digital advice and real-world medical assistance.

5.7 Web Application Deployment using Flask

The system is deployed as a web-based application using Flask.

Features:

- Chat interface
- Message input field
- Real-time response rendering
- Backend REST API for model inference

Communication Flow:

User → Web UI → Flask Backend → Chatbot Engine → Response → UI Display

5.8 Testing and Validation

The system is evaluated using real symptom-based queries.

Metrics:

- Intent classification accuracy
- Response relevance
- Response time
- System stability
- Error handling efficiency

Edge cases and unknown inputs are handled using fallback responses.

5.9 Final System Characteristics

- Real-time conversational AI
- Lightweight ML model
- Scalable intent structure
- Web accessible
- Location-aware
- Extendable dataset

CHAPTER 6

REQUIREMENTS

6.1 Functional Requirements

The functional requirements describe the core operations that the system must perform to fulfill its intended purpose.

1. The system shall allow users to enter health-related queries and symptoms using natural language through a web-based interface.
2. The system shall preprocess user input using NLP techniques including tokenization, normalization, stemming, and vectorization.
3. The system shall classify user queries into predefined intent categories using a trained RNN model.
4. The system shall generate appropriate responses based on the predicted intent from the knowledge base.
5. The system shall detect low-confidence predictions and provide fallback responses for unclear inputs.
6. The system shall retrieve and display nearby medical centers when medical consultation is required.
7. The system shall support real-time interaction between the user and the chatbot.
8. The system shall allow easy extension of the dataset by adding new intents and responses.
9. The system shall store and load trained model parameters for reuse without retraining.
10. The system shall handle invalid inputs and system errors gracefully.

6.2 Non-Functional Requirements

The non-functional requirements specify the quality attributes and performance expectations of the system.

1. **Performance:**

   The chatbot should respond to user queries within 2 seconds to ensure smooth interaction.

2. **Scalability:**
   The system should support future expansion in terms of dataset size and number of users.
3. **Reliability:**
   The chatbot should operate continuously without frequent crashes or failures.
4. **Usability:**
   The interface should be simple, intuitive, and accessible to non-technical users.
5. **Maintainability:**
   The system should be modular, allowing easy updates to the model, dataset, and UI.
6. **Portability:**
   The system should run on different operating systems with minimal configuration.
7. **Security:**
   User inputs should not be stored unnecessarily and must be handled securely.
8. **Accuracy:**
   The intent classification accuracy should be sufficiently high to ensure meaningful responses.

6.3 Software Requirements

The following software components are required for development and execution of the system:

Operating System

- Windows 10 / 11, Linux, or macOS

Programming Language

- Python 3.8 or higher

Frameworks and Libraries

- Flask – Web application framework
- PyTorch – Neural network implementation
- NLTK – Natural Language Processing
- NumPy – Numerical operations
- Geocoder – Location detection
- JSON – Dataset storage
- HTML, CSS, JavaScript – Frontend interface

Development Tools

- VS Code / PyCharm
- Web Browser (Chrome / Firefox)
- Git (Version control)

6.4 Hardware Requirements

The system can run on standard consumer hardware.

Minimum Requirements

- Processor: Intel i3 or equivalent
- RAM: 4 GB
- Storage: 2 GB free disk space
- Internet connection (optional, for location services)

Recommended Requirements

- Processor: Intel i5 or equivalent
- RAM: 8 GB or higher
- Storage: 5 GB free disk space
- GPU (optional): For faster model training
- Stable internet connection

CHAPTER 7

DESIGN

The system design of the Symptom Checker Chatbot defines the overall structure, components, data flow, and interaction between different modules of the application. The design follows a modular and layered architecture to ensure scalability, maintainability, and efficient real-time performance.

The system is divided into three main layers: the User Interface Layer, the Application Processing Layer, and the Machine Learning Layer.

7.1 Overall Architecture Design

The chatbot system follows a client–server architecture where:

- The client interacts through a web browser.
- The server processes requests using Flask.
- The machine learning model performs intent classification and response selection.

High-Level Architecture Components:

1. Web-based User Interface
2. Flask Backend Server
3. NLP Processing Module
4. RNN Intent Classification Model
5. Knowledge Base (intents.json)
6. Medical Center Database
7. Response Generator

7.2 User Interface Design

The user interface is designed to be simple and interactive.

Features:

- Chat window to display conversation history
- Input text box for entering symptoms
- Send button to submit queries
- Automatic scrolling and response display

The UI is implemented using HTML, CSS, and JavaScript and communicates with the backend using HTTP requests.

7.3 Backend Design (Flask Server)

The Flask server acts as the central controller of the system.

Responsibilities:

- Receive user messages from UI
- Forward messages to NLP module
- Invoke ML model for prediction
- Fetch responses from knowledge base
- Append medical center data when required
- Return formatted response to UI

The backend exposes REST endpoints to handle chatbot requests.

7.4 NLP Processing Module Design

This module converts human language into numerical format.

Internal Components:

- Tokenizer
- Stemmer
- Vocabulary builder
- Bag-of-Words vector generator

It ensures consistency between training and runtime inference.

7.5 Machine Learning Model Design

The RNN-based classifier is responsible for understanding user intent.

Design Characteristics:

- Lightweight architecture
- Fast inference speed
- Multi-class intent classification
- Confidence-based prediction filtering

The trained model is stored as data_rnn.pth and loaded during runtime.

7.6 Knowledge Base Design

The knowledge base is implemented as a structured JSON file.

Structure:

- Intent Tag

- List of Patterns
- List of Responses

This allows easy extension by adding new intents without modifying code logic.

7.7 Medical Center Recommendation Module Design

This module enhances system practicality.

Design Flow:

1. Detect user location
2. Load medical center dataset
3. Filter nearest facilities
4. Attach recommendations to chatbot reply

CHAPTER 8

RESULTS AND DISCUSSIONS

This chapter presents the outcomes obtained from the implementation of the Symptom Checker Chatbot and discusses the system's performance, usability, and effectiveness in providing preliminary healthcare assistance.

8.1 Experimental Setup

The system was tested on a local machine using the trained RNN-based intent classification model integrated with the Flask web application. Various symptom-related queries, general health questions, and emergency-type inputs were used to evaluate the chatbot. The evaluation focused on intent recognition accuracy, response relevance, system responsiveness, and overall user experience.

8.2 Results

8.2.1 Intent Classification Performance

The trained RNN model demonstrated reliable performance in classifying user inputs into predefined intent categories. For common symptom queries such as fever, headache, cough, fatigue, and stomach pain, the chatbot consistently predicted the correct intent and returned relevant responses.

- The model achieved high accuracy for frequently occurring symptom patterns.
- The confidence threshold mechanism successfully filtered ambiguous inputs and triggered fallback responses when necessary.
- Misclassifications were minimal and mostly occurred for highly complex or uncommon symptom descriptions.

8.2.2 Response Quality

The responses generated by the chatbot were:

- Contextually relevant to the user's query
- Clear and easy to understand
- Informational rather than diagnostic

When medical assistance was required, the chatbot correctly appended nearby medical center details, improving the practical usefulness of the system.

8.2.3 System Responsiveness

The average response time for user queries was under 2 seconds, ensuring smooth real-time interaction. The lightweight RNN model and optimized preprocessing pipeline contributed to low latency and stable performance even after multiple consecutive requests.

8.2.4 Medical Center Recommendation Accuracy

The location-based module successfully retrieved relevant healthcare facilities based on the user's approximate location. The displayed information included the medical center name and location details, enabling users to take immediate action if required.

8.2.5 User Interface Evaluation

The web-based interface was found to be:

- Simple and intuitive
- Easy to navigate
- Suitable for non-technical users

The chat layout enabled continuous interaction without page reloads, enhancing user experience.

8.3 Discussion

The results indicate that the Symptom Checker Chatbot effectively fulfills its primary objective of providing preliminary healthcare guidance through natural language interaction. The use of an RNN-based intent classifier combined with NLP preprocessing enabled accurate interpretation of user queries and appropriate response generation.

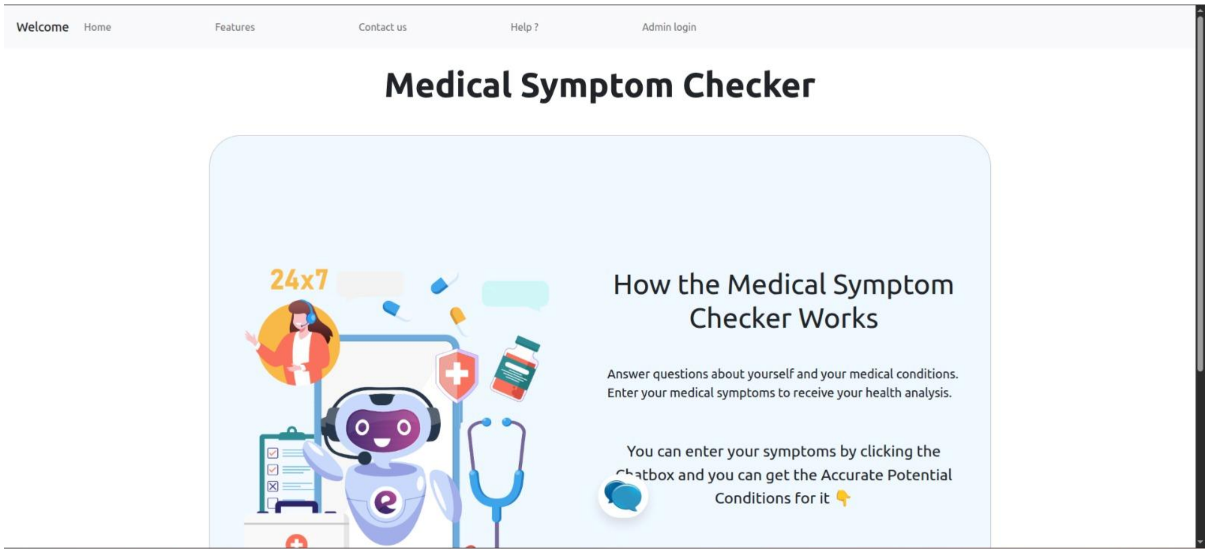
The modular design allowed seamless integration of the machine learning model, knowledge base, and medical center recommendation system. This design choice significantly improved system maintainability and future scalability.

However, certain limitations were observed:

- The chatbot's knowledge is restricted to the intents present in the dataset.
- Complex multi-symptom medical cases may not always be interpreted accurately.
- The system does not perform clinical diagnosis and should not be treated as a substitute for professional medical advice.

Despite these limitations, the chatbot demonstrated strong potential as a first-level healthcare assistance tool. Its low computational requirements, real-time performance, and ease of deployment make it suitable for educational use and as a foundation for more advanced healthcare AI systems.

8.4 Qualitative Results



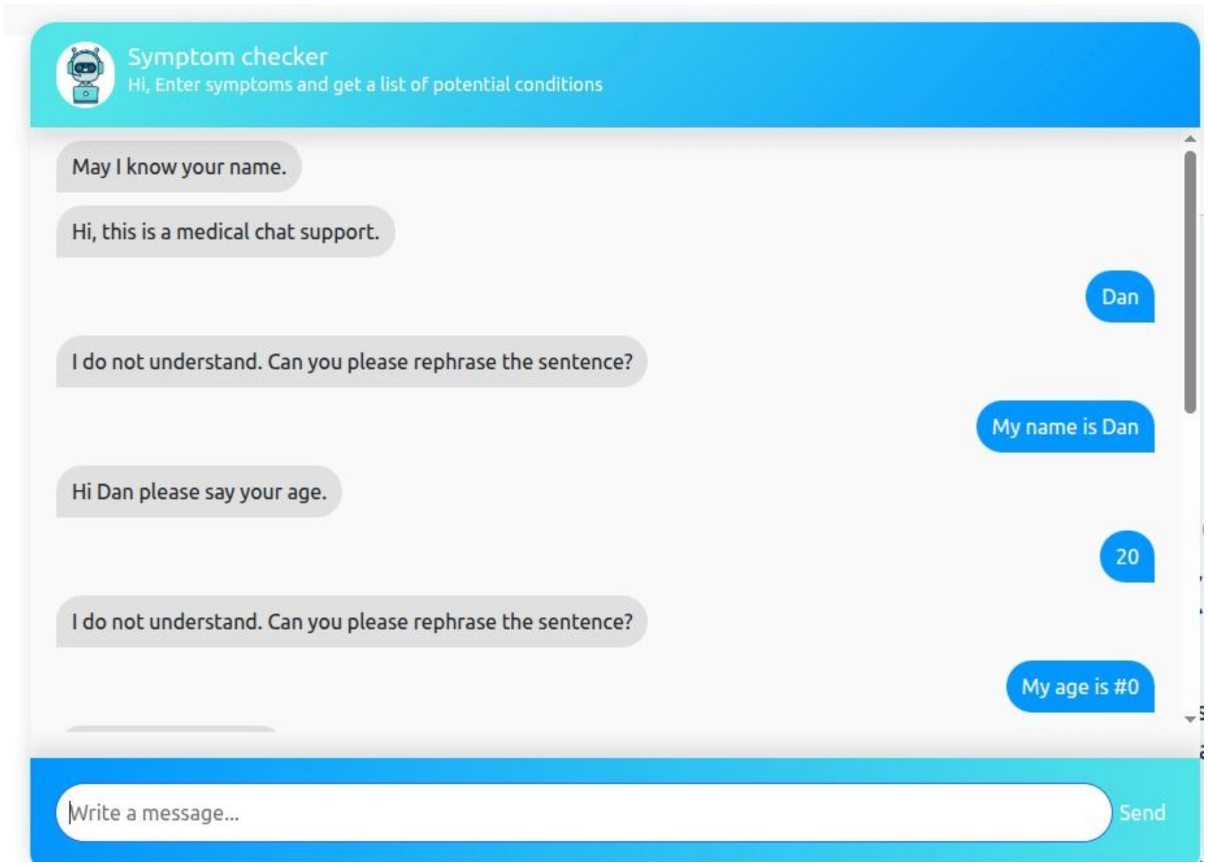
Medical Symptom Checker

How the Medical Symptom Checker Works

Answer questions about yourself and your medical conditions. Enter your medical symptoms to receive your health analysis.

You can enter your symptoms by clicking the Chatbox and you can get the Accurate Potential Conditions for it 👇

Figure 8.4.1: Web Page Result


Symptom checker
Hi, Enter symptoms and get a list of potential conditions

May I know your name.

Hi, this is a medical chat support.

Dan

I do not understand. Can you please rephrase the sentence?

My name is Dan

Hi Dan please say your age.

20

I do not understand. Can you please rephrase the sentence?

My age is #0

Figure 8.4.2: Chatbot Result

CHAPTER 9

CONCLUSION

The Symptom Checker Chatbot project successfully demonstrates the effective application of artificial intelligence, natural language processing, and web-based technologies to develop an intelligent, real-time healthcare assistance system. The system accurately interprets user-reported symptoms, classifies them into predefined intents using an RNN-based model, and generates contextually relevant responses while providing information about nearby medical centers when necessary. Its modular and scalable design allows easy maintenance, dataset expansion, and integration of additional functionalities in the future. The web interface is intuitive and user-friendly, ensuring accessibility for users with minimal technical knowledge, while the lightweight model and optimized preprocessing ensure fast, real-time interactions. Although the system is not a substitute for professional medical diagnosis, it serves as an effective first-level healthcare guidance tool, bridging the gap between patients and medical facilities. Overall, the project highlights the potential of AI-driven conversational systems in enhancing healthcare awareness, accessibility, and preliminary decision support, laying a foundation for further improvements in intelligent healthcare solutions.