VISVESVARAYA TECHNOLOGICAL UNIVERSITY,

BELAGAVI - 590018

Mini Project Report

On

"FIRST AID CHATBOT"

A report submitted in partial fulfillment of the requirements for

MINI PROJECT (21AIMP67)
In
Artificial Intelligence & Machine Learning

Submitted by

ABHINAV ASHOK	4AL21AI001
MANOHARA M	4AL21AI023
NITHIN	4AL21AI027
VJ JISON	4AL21AI057

Under the Guidance of

Mr. Kiran Raj K M

Assistant Professor

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY MIJAR,
(Unit of Alva's Education Foundation ®, Moodbidri)
Affiliated to Visvesvaraya Technological University, Belagavi,
Approved by AICTE, New Delhi, Recognized by Government of Karnataka.
Accredited by NACC with A+ Grade
Shobavana Campus, Mijar, Moodbidri, D.K., Karnataka
2023 – 2024

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY
MIJAR, MOODBIDRI, D.K. -574225

ALVA'S
Education Foundation®

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

CERTIFICATE

This is to certify that the Mini Project entitled **"FIRST AID CHATBOT"** has been successfully completed by

ABHINAV ASHOK	4AL21AI001
MANOHARA M	4AL21AI023
NITHIN	4AL21AI027
VJ JISON	4AL21AI057

The Bonafide students of the Department of Artificial Intelligence and Machine Learning, Alva's Institute of Engineering and Technology in the **DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING** of the VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI during the year 2023–2024. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Mini Project report has been approved as it satisfies the academic requirements in respect of the Mini Project work prescribed for the Bachelor of Engineering Degree.

Mr. Kiran Raj K M	**Dr. Pradeep Nazareth**	**Prof. Harish Kunder**
Project Guide	**Project Coordinator**	**HOD, Dept. of AIML**

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY
MIJAR, MOODBIDRI D.K. -574225

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

DECLARATION

We,

ABHINAV ASHOK
MANOHARA M
NITHIN
VJ JISON

Hereby declare that the dissertation entitled, **"FIRST AID CHATBOT"** is completed and written by us under the supervision of my guide **Mr. Kiran Raj K M**, **Assistant Professor,** Department of Artificial Intelligence and Machine Learning Alvas's Institute of Engineering and Technology, Moodbidri, **DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHIME LEARNING** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI** during the academic year 2023-2024. The dissertation report is original and it has not been submitted for any other degree in any university.

ABHINAV ASHOK	**4AL21AI001**
MANOHARA M	**4AL21AI023**
NITHIN	**4AL21AI027**
VJ JISON	**4AL21AI057**

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of the people who made it possible, success is the epitome of hardwork and perseverance, but steadfast of all is encouraging guidance.

So, with gratitude, we acknowledge all those whose guidance and encouragement served as a beacon of light and crowned the effort with success.

The selection of this mini-project work as well as the timely completion is mainly due to the interest and persuasion of our Project guide **Mr. Kiran Raj K M,** Assistant Professor, Department of Artificial Intelligence & Machine Learning. We will remember his contribution forever.

We sincerely thank our Project Coordinator **Dr. Pradeep Nazareth,** Associate Professor, Department of Artificial Intelligence & Machine Learning for his Constant Support.

We sincerely thank, **Prof. Harish Kunder**, Associate Professor and Head of the Department of Artificial Intelligence & Machine Learning who has been the constant driving force behind the completion of the project.

We thank our beloved Principal, **Dr. Peter Fernandes,** for his constant help and support throughout.

We are indebted to the **Management of Alva's Institute of Engineering and Technology, Mijar, Moodbidri** for providing an environment which helped us in completing our mini project.

Also, we thank all the teaching and non-teaching staff of Department of Artificial Intelligence & Machine Learning for the help rendered.

ABHINAV ASHOK	**4AL21AI001**
MANOHARA M	**4AL21AI023**
NITHIN	**4AL21AI027**
VJ JISON	**4AL21AI057**

ABSTRACT

We started this project to provide immediate first aid information during natural disasters, using NLP and machine learning to offer quick and reliable guidance. The First Aid Chatbot provides quick, reliable first aid information during natural disasters. Using Natural Language Processing (NLP) and machine learning, it understands user queries and offers relevant first aid advice. The chatbot uses Natural Language Toolkit (NLTK) for tokenizing and tagging inputs, and the Levenshtein distance algorithm to find the best match for the user's query in its database. Built with Flask, it features an easy-to-use web interface. This tool aims to deliver lifesaving information promptly, improving survival rates and health outcomes during disasters by offering immediate first aid guidance and reducing the burden on emergency services.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In times of disaster, immediate and effective first aid can be the difference between life and death. Natural and man-made disasters often result in numerous injuries, and the ability to provide quick, accurate first aid is crucial. However, the sheer scale of such events can overwhelm medical personnel and delay professional medical assistance. This project aims to address this critical need by developing a First Aid Chatbot. This innovative tool leverages advanced predictive algorithms and natural language processing to provide instant, reliable first aid guidance based on user input.

The chatbot is designed to be user-friendly, supporting text interactions to accommodate various user preferences and needs. By providing accessible, accurate, and immediate first aid information, this project enhances disaster response efforts and contributes to better outcomes for those affected by disasters [1]. By interacting with the chatbot, users can receive tailored advice on how to manage injuries and other medical emergencies, bridging the gap until professional help arrives.

1.1 PROBLEM STATEMENT

The primary problem addressed by this project is the lack of immediate, accessible, and accurate first aid guidance during disaster situations. Traditional first aid resources, such as manuals or waiting for medical personnel, may not be feasible in urgent scenarios where every second counts [2]. The project aims to develop a chatbot that can predict and provide necessary first aid instructions based on the specific conditions described by the user, thereby enhancing the chances of survival and reducing the impact of injuries during disasters.

1.2 OBJECTIVES

The project focuses on creating a user-friendly chatbot that helps people during disaster situations. The interface is simple and easy to use, so even in stressful times, users can quickly navigate and get the help they need. Clear instructions and a clean layout make it easy to input symptoms and receive guidance without any trouble. The chatbot uses advanced algorithms to understand user inputs and provide accurate first aid advice based on the symptoms described. It follows established first aid protocols and has been tested and refined to ensure the advice is reliable and precise. Additionally, the chatbot supports text-based interactions, allowing users to type their questions and get responses.

1.2.1 USER FRIENDLY INTERFACE

To develop a user-friendly chatbot interface that allows individuals to easily interact with it during disaster situations. The project successfully created an intuitive and accessible interface for the chatbot [3]. The design focuses on simplicity and ease of use, ensuring that even in high-stress situations, users can quickly navigate the chatbot. Clear prompts, straightforward navigation, and a clean layout make it easy for users to input their symptoms and receive guidance without any technical difficulties.

1.2.2 ACCURATE FIRST AID SUGGESTION

To implement predictive algorithms that analyse user inputs and provide accurate first aid guidance tailored to the described symptoms and situations. The chatbot uses advanced algorithms to process the information provided by users and predict the most appropriate first aid steps [4]. By leveraging a comprehensive dataset and adhering to established first aid protocols, the chatbot ensures that the suggestions given are precise and reliable. Continuous testing and refinement have further enhanced the accuracy of these predictions, providing users with dependable advice.

1.2.3 TEXT CONVERSATION

To support text-based interactions, allowing users to communicate with the chatbot in their preferred mode, the project incorporated functionality for text-based interactions. This ensures that users can interact with the chatbot through typing, making the system versatile and accessible in various scenarios. Natural language processing technology was employed to facilitate smooth and accurate text conversations.

1.3 SCOPE OF THE PROJECT

- ➢ The project will cover the development of a chatbot that interacts with users through a conversational interface, capable of understanding and processing natural language inputs related to first aid queries.
- ➢ The project will involve testing and validating the chatbot to ensure its accuracy and reliability in providing first aid advice.
- ➢ It will include a user interface that is accessible via web browsers.

CHAPTER 2

LITERATURE REVIEW

S. Anushka and S. Thelijjagoda, "Healbot: NLP-based Health Care Assistant for Global Pandemics", this paper presents "Healbot," a healthcare assistant designed using natural language processing (NLP) to support global pandemic response efforts [1]. The system aims to provide accurate and timely health information, facilitate self-diagnosis, and assist with mental health support during pandemics. The authors discuss the architecture, implementation, and performance of Healbot, highlighting its potential impact on improving healthcare accessibility and management during crises.

Sreedhar Kumar S, Syed Thouheed Ahmed, Afifa Salsabil Fathima, Nishabai M and Sophia S, "Medical ChatBot Assistance for Primary Clinical Guidance using Machine Learning Techniques, this study introduces a medical chatbot that provides primary clinical guidance using machine learning techniques [2]. The chatbot is designed to assist patients in obtaining preliminary medical advice based on symptoms, thus reducing the burden on healthcare professionals. The paper details the machine learning models employed, the chatbot's training process, and its effectiveness in delivering accurate medical guidance.

K. Arumugam, Mohd Naved, Priyanka P. Shinde, Orlando Leiva-Chauca, Antonio Huaman-Osorio and Tatiana Gonzales-Yanac, "Multiple disease prediction using Machine learning algorithms", the authors discuss a system that uses machine learning algorithms to predict multiple diseases based on patient data [3]. The paper covers various algorithms, their implementation, and comparative performance analysis. The goal is to enhance early diagnosis and treatment planning by leveraging data-driven insights, ultimately improving patient outcomes and healthcare efficiency.

Yang and Christopher C, "Explainable Artificial Intelligence for Predictive Modeling in Healthcare", this paper explores the application of explainable artificial intelligence (XAI) in predictive modeling within the healthcare sector [4]. Christopher C. Yang emphasizes the importance of transparency and interpretability in AI models used for healthcare decisions. The paper discusses methods to make AI models more explainable and the potential benefits for clinicians, patients, and regulatory bodies in understanding and trusting AI-driven healthcare solutions.

Shashidhar, R. Patilkulkarni, S., Puneeth and S. B., "Combining audio and visual speech recognition using LSTM and deep convolutional neural network", this research combines audio and visual speech recognition technologies using Long Short-Term Memory (LSTM) networks and deep convolutional neural networks (CNNs) [5]. The authors present a hybrid model designed to improve speech recognition accuracy by integrating audio and visual data. The paper details the model architecture, training process, and performance evaluation, demonstrating the advantages of multimodal speech recognition systems.

Chou, J. S., Chong, P. L., Liu and C. Y. "Deep learning-based chatbot by natural language processing for supportive risk management in river dredging projects", the authors introduce a deep learning-based chatbot that employs natural language processing (NLP) for risk management in river dredging projects [6]. The chatbot assists project managers by providing real-time risk assessments, suggestions, and support. The paper outlines the chatbot's development, its integration into the project management workflow, and its effectiveness in enhancing decision-making and reducing project risks.

Ouerhani, N., Maalel, A., Ben Ghézela and H., "SPeCECA: a smart pervasive chatbot for emergency case assistance based on cloud computing", this study presents SPeCECA, a smart pervasive chatbot designed for emergency case assistance using cloud computing [7]. The chatbot aims to provide immediate support and guidance during emergencies by leveraging cloud-based resources for real-time data processing and response generation. The authors describe the system's architecture, implementation, and potential benefits in improving emergency response efficiency and effectiveness.

Ayanouz, S., Abdelhakim, B. A., Benhmed and M., "A smart chatbot architecture based NLP and machine learning for health care assistance", The paper discusses a smart chatbot architecture that integrates NLP and machine learning for healthcare assistance [8]. The chatbot is designed to provide personalized medical advice, manage patient inquiries, and facilitate healthcare services. The authors explain the system's design, the underlying technologies, and its performance in real-world scenarios, highlighting its potential to enhance healthcare delivery.

Soufyane, A., Abdelhakim, B. A., Ahmed and M. B., "An intelligent chatbot using NLP and TF-IDF algorithm for text understanding applied to the medical field", this paper describes an intelligent chatbot that utilizes NLP and the Term Frequency-Inverse Document Frequency (TF-IDF) algorithm for understanding medical texts [9]. The chatbot aims to assist healthcare professionals and patients by accurately interpreting medical information and providing relevant responses. The authors detail the chatbot's development, the implementation of NLP and TF-IDF, and the system's effectiveness in the medical domain.

Mokmin, N. A. M., Ibrahim and N. A., “The evaluation of chatbot as a tool for health literacy education among undergraduate students”, the study evaluates the effectiveness of a chatbot as a tool for health literacy education among undergraduate students [10]. The authors explore how the chatbot facilitates learning, enhances health knowledge, and engages students. The paper discusses the chatbot's design, educational content, and assessment of its impact on students' health literacy and overall learning experience.

Ouerhani, N., Maalel, A., Ben Ghézala and H., “SMAD: SMart assistant during and after a medical emergency case based on deep learning sentiment analysis: The pandemic COVID-19 case”, this research introduces SMAD, a smart assistant designed for use during and after medical emergencies, employing deep learning sentiment analysis [11]. The system was developed in response to the COVID-19 pandemic, aiming to provide emotional support and guidance to patients and healthcare workers. The authors describe the assistant's architecture, sentiment analysis capabilities, and its role in managing the emotional aspects of medical emergencies.

Ryu, H., Kim, S., Kim, D., Han, S., Lee, K., Kang and Y., “Simple and steady interactions win the healthy mentality: designing a chatbot service for the elderly”, this paper focuses on designing a chatbot service tailored for the elderly to promote mental health and well-being [12]. The authors emphasize the importance of simple and steady interactions to maintain a healthy mentality among older adults. The paper discusses the design principles, user interface considerations, and the chatbot's impact on the mental health of elderly users, highlighting its potential to provide companionship and support.

CHAPTER 3

SYSTEM ANALYSIS

3.1 INTRODUCTION

In the context of disaster response, the need for timely and accurate first aid information is paramount. A First Aid Chatbot aims to address this need by providing immediate assistance through a user-friendly interface. The system is designed to leverage advanced technologies such as natural language processing (NLP) and machine learning to understand user queries related to various disaster scenarios and offer relevant first aid advice. By integrating real-time data, the chatbot will ensure accessibility and reliability, enhancing the overall effectiveness of disaster response efforts [2]. Disasters, whether natural or man-made, often result in chaos and confusion, making it challenging for individuals to access accurate first aid information quickly. Traditional methods of disseminating first aid advice, such as manuals or even online resources, may not be readily available or practical in such critical situations [6]. This is where the First Aid Chatbot comes into play, offering a seamless and immediate solution to deliver crucial medical guidance directly to those in need.

The chatbot's ability to process natural language allows it to interact with users in a conversational manner, making it easier for individuals to describe their situations and receive tailored advice. For instance, during an earthquake, a user might inquire about how to treat a specific type of injury, and the chatbot, utilizing its machine learning algorithms, can provide step-by-step instructions based on the latest medical guidelines. This not only improves the user experience but also ensures that the advice given is relevant to the specific circumstances described by the user [7]. Moreover, the system's integration with real-time data sources ensures that the information it provides is current and accurate. This is particularly important in disaster scenarios where the situation can change rapidly, and outdated information could be harmful. By constantly updating its database with the latest medical and situational data, the chatbot maintains a high level of reliability and trustworthiness.

3.2 EXISTING SYSTEM

The current solutions available for first aid guidance during disasters are often limited and not specialized. Most existing systems lack the ability to provide real-time, context-specific advice tailored to the unique circumstances of different disasters. They typically offer generalized information that may not be immediately applicable to the user's situation. Additionally, these systems may not support multiple languages, restricting their accessibility to non-English speaking populations. The integration with up-to-date medical databases is often inadequate, leading to outdated or incorrect information being disseminated.

3.3 PROPOSED SYSTEM

The proposed First Aid Chatbot will address the shortcomings of existing systems by offering a specialized, AI-driven solution. Utilizing machine learning and NLP, the chatbot will accurately interpret user inputs and provide precise first aid measures tailored to specific disaster scenarios [5]. It will be accessible via web applications, ensuring broad reach and usability. Real-time data integration will keep the information current and reliable, while a database of user interactions will allow for continuous improvement of the system's responses.

3.4 FEATURES OF SOFTWARE

In the project, Python with Flask is used for the backend to handle server-side tasks like processing user inputs and running predictive algorithms. Flask is a lightweight web framework that makes developing web applications easy. For the frontend, HTML is used to structure the web pages, while CSS is used to style them. HTML defines the layout with headings, paragraphs, images, and links, and CSS controls the look, including colors, fonts, and layout. This combination ensures that the chatbot has a smooth, efficient backend and a visually appealing, user-friendly frontend.

3.4.1 PYTHON-FLASK BACKEND

Python with Flask is used for the backend of our project. Flask is a lightweight web framework for Python that facilitates the development of web applications. In your project, Flask handles server-side logic, such as processing user inputs, executing predictive algorithms, and generating dynamic responses [8]. It integrates seamlessly with HTML templates to render pages dynamically based on user interactions, ensuring that the chatbot functions smoothly and efficiently.

3.4.2 HTML-CSS FRONTEND

HTML (Hypertext Markup Language) is the standard markup language for creating web pages. It uses tags to define the structure and content of a web page, such as headings, paragraphs, images, and links. In your project, HTML is used to create the frontend interface that users interact with, presenting information and forms in a structured manner. CSS (Cascading Style Sheets) is a styling language that complements HTML by defining the visual presentation of web pages [3]. It allows developers to control layout, colors, fonts, and other design elements across multiple pages. CSS separates the presentation from the structure defined in HTML, enhancing the aesthetics and consistency of your project's frontend. It ensures a visually appealing and user-friendly interface by applying styles to HTML elements.

3.5 SOFTWARE REQUIREMENTS

Operating System	: Windows 10/11
Browser	: Microsoft-Edge/Google Chrome
Dataset	: JSON File
Technology	: Python, Flask, HTML, CSS

3.6 HARDWARE REQUIREMENTS

Hard Disk Drive	: 500GB
Processor	: 1.8GHz
RAM	: 4GB

CHAPTER 4

METHODOLOGY

4.1 PROJECT DESIGN

The chatbot is designed as a web application using the Python Flask framework for the backend and HTML/CSS/JavaScript for the frontend. The architecture follows a client-server model where user inputs are processed on the server side. The backend integrates with a JSON file containing a dataset of predefined symptoms and corresponding first aid responses [9]. When a user inputs symptoms, the backend tokenizes the input, searches for matching symptoms in the JSON dataset, and outputs the corresponding first aid instructions. The chatbot interface includes options for text input and potentially voice input, designed for intuitive user interaction during disaster scenarios.

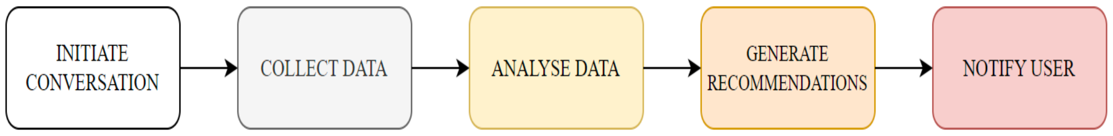


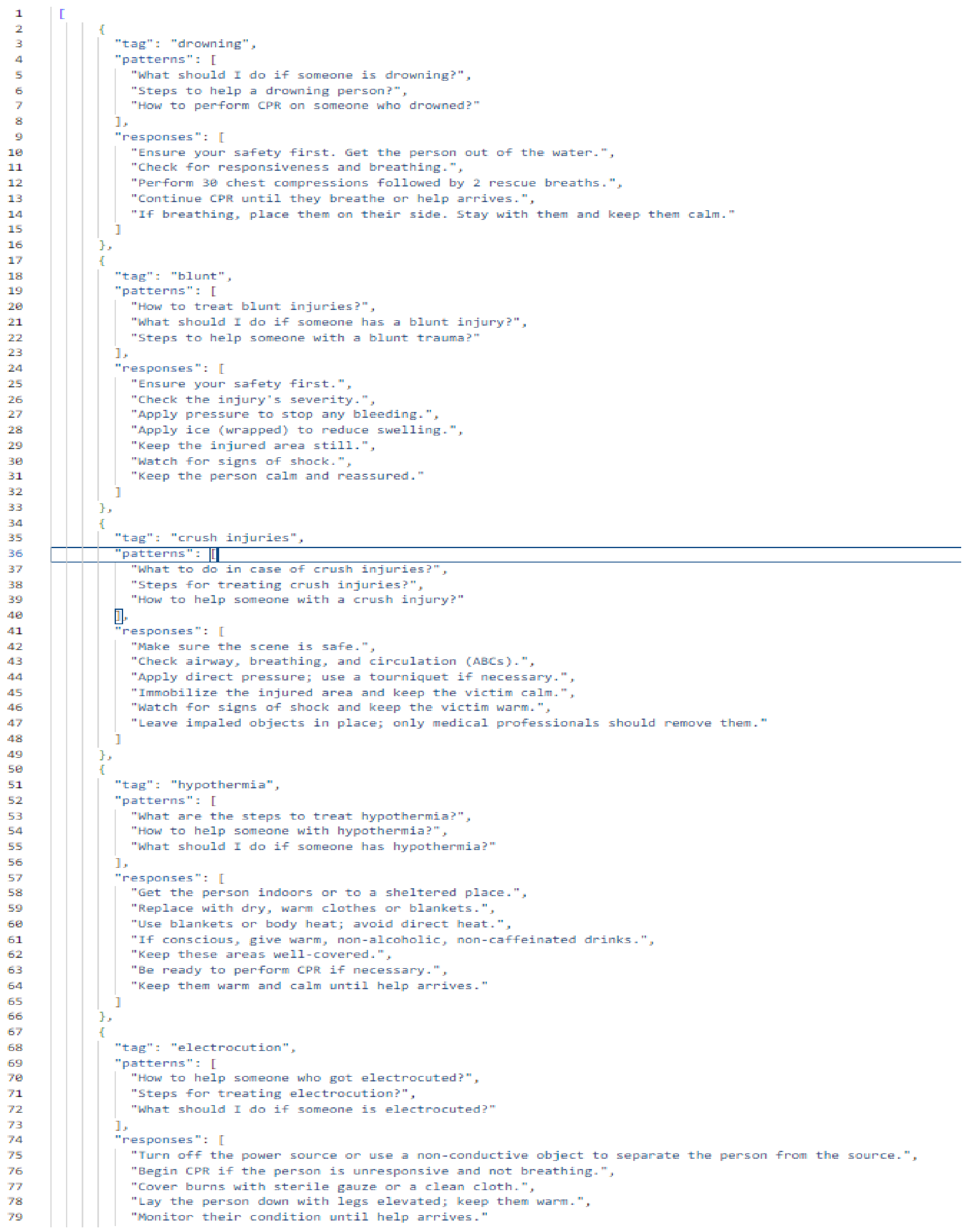
Figure 4.1 PROJECT DESIGN

Figure 4.1 illustrates the workflow of the First Aid Chatbot, which begins with the user initiating a conversation by describing their symptoms. The chatbot then collects detailed information from the user, ensuring all relevant data is gathered. Next, the collected data is analyzed using tokenization and matching algorithms, including the Levenshtein distance algorithm, to identify the most relevant first aid instructions [10]. Based on this analysis, the system generates specific first aid recommendations tailored to the user's symptoms. Finally, these recommendations are communicated back to the user through the chatbot interface, providing clear and actionable first aid guidance in real-time.

4.2 DATA COLLECTION

Data for the chatbot's JSON dataset were curated from authoritative medical resources, disaster response guidelines, and relevant academic literature. This rigorous compilation ensures that the dataset is both comprehensive and reliable. The dataset includes a wide array of common disaster scenarios, their associated symptoms, and the corresponding recommended first aid responses. To further enhance the accuracy and reliability of the dataset, real-world incident data may have been incorporated, allowing for validation and refinement of the symptom-response mappings.

Figure 4.2 presents the collected dataset obtained from Kaggle, an open data platform known for hosting datasets from various domains. The dataset sourced from Kaggle was meticulously selected to complement the curated data, providing additional depth and breadth. The integration of this dataset into the JSON file was carefully handled to maintain consistency and accuracy in the symptom-response mappings. This step ensures that the chatbot is equipped with a robust and validated dataset capable of delivering accurate first aid recommendations in diverse disaster scenarios.



```
[
    {
      "tag": "drowning",
      "patterns": [
        "What should I do if someone is drowning?",
        "Steps to help a drowning person?",
        "How to perform CPR on someone who drowned?"
      ],
      "responses": [
        "Ensure your safety first. Get the person out of the water.",
        "Check for responsiveness and breathing.",
        "Perform 30 chest compressions followed by 2 rescue breaths.",
        "Continue CPR until they breathe or help arrives.",
        "If breathing, place them on their side. Stay with them and keep them calm."
      ]
    },
    {
      "tag": "blunt",
      "patterns": [
        "How to treat blunt injuries?",
        "What should I do if someone has a blunt injury?",
        "Steps to help someone with a blunt trauma?"
      ],
      "responses": [
        "Ensure your safety first.",
        "Check the injury's severity.",
        "Apply pressure to stop any bleeding.",
        "Apply ice (wrapped) to reduce swelling.",
        "Keep the injured area still.",
        "Watch for signs of shock.",
        "Keep the person calm and reassured."
      ]
    },
    {
      "tag": "crush injuries",
      "patterns": [
        "What to do in case of crush injuries?",
        "Steps for treating crush injuries?",
        "How to help someone with a crush injury?"
      ],
      "responses": [
        "Make sure the scene is safe.",
        "Check airway, breathing, and circulation (ABCs).",
        "Apply direct pressure; use a tourniquet if necessary.",
        "Immobilize the injured area and keep the victim calm.",
        "Watch for signs of shock and keep the victim warm.",
        "Leave impaled objects in place; only medical professionals should remove them."
      ]
    },
    {
      "tag": "hypothermia",
      "patterns": [
        "What are the steps to treat hypothermia?",
        "How to help someone with hypothermia?",
        "What should I do if someone has hypothermia?"
      ],
      "responses": [
        "Get the person indoors or to a sheltered place.",
        "Replace with dry, warm clothes or blankets.",
        "Use blankets or body heat; avoid direct heat.",
        "If conscious, give warm, non-alcoholic, non-caffeinated drinks.",
        "Keep these areas well-covered.",
        "Be ready to perform CPR if necessary.",
        "Keep them warm and calm until help arrives."
      ]
    },
    {
      "tag": "electrocution",
      "patterns": [
        "How to help someone who got electrocuted?",
        "Steps for treating electrocution?",
        "What should I do if someone is electrocuted?"
      ],
      "responses": [
        "Turn off the power source or use a non-conductive object to separate the person from the source.",
        "Begin CPR if the person is unresponsive and not breathing.",
        "Cover burns with sterile gauze or a clean cloth.",
        "Lay the person down with legs elevated; keep them warm.",
        "Monitor their condition until help arrives."
```

Figure 4.2 DATASET

4.3 DATA PROCESSING

- Data Compilation: Symptoms and corresponding first aid responses were compiled into a structured JSON format.
- Tokenization: User inputs are tokenized into individual symptom tokens using natural language processing techniques.
- Matching and Retrieval: The Levenshtein distance algorithm calculates the similarity between user inputs and predefined first aid tags to ensure the best possible response.
- Output Generation: Based on matched tokens, the backend retrieves corresponding first aid responses from the JSON file for output to the user.

4.4 MODEL DEVELOPMENT

Given the token-based approach, the project focuses on efficient search and retrieval mechanisms rather than traditional machine learning models. The system uses algorithms to tokenize user inputs and efficiently search through the JSON dataset to retrieve accurate first aid instructions [11]. This approach leverages rapid token matching and retrieval techniques to ensure real-time responses without the need for extensive model training.

4.4.1 TOKENIZATION PROCESS

The tokenization process involves breaking down user input into smaller units, called tokens, which can be individual words or phrases. These tokens are then used to match entries in the JSON dataset. Tokenization helps in handling variations in user input by focusing on essential keywords and phrases, making the search process more robust and accurate.

4.4.2 SEARCH AND RETRIEVAL ALGORITHM

The search and retrieval algorithm plays a critical role in the system. It is designed to quickly match tokens from the user input with the tokens in the JSON dataset. The algorithm uses the following steps.

- Token Extraction: Extract tokens from the user input using natural language processing (NLP) techniques.
- Token Matching: Compare extracted tokens with tokens in the JSON dataset.
- Relevance Scoring: Calculate relevance scores based on the number of matching tokens and their significance.
- Result Retrieval: Retrieve the most relevant first aid instructions based on the highest relevance scores.

- This method ensures that the system provides accurate and contextually appropriate first aid instructions in real-time.

4.4.3 JSON DATASET STRUCTURE

The JSON dataset is structured to facilitate efficient search and retrieval. Each entry in the dataset contains:

- Condition: A brief description of the medical condition or situation.
- Symptoms: A list of symptoms associated with the condition.
- Instructions: Detailed first aid instructions for addressing the condition.
- Tokens: Pre-computed tokens for each entry to speed up the search process.

By organizing the data in this manner, the system can quickly access and retrieve relevant information without extensive computation.

4.4.4 ADVANTAGES OF THE TOKEN-BASED APPROACH

The token-based approach offers several advantages over traditional machine learning models.

- Speed: Rapid token matching and retrieval techniques ensure real-time responses.
- Simplicity: Eliminates the need for extensive model training and maintenance.
- Robustness: Handles variations in user input effectively by focusing on essential keywords.
- Scalability: Easily scalable as the JSON dataset can be expanded without significant changes to the algorithm.

4.5 IMPLEMENTATION

The implementation phase involves setting up the backend, integrating the JSON dataset, and developing the frontend to create a seamless user experience for the First Aid Chatbot. The following sections detail the steps involved in this phase.

4.5.1 BACKEND DEVELOPMENT

The backend server is developed using Python Flask, a lightweight and flexible web framework [12]. Flask is chosen for its simplicity and ease of integration with other technologies. The backend development involves following phases.

- Server Setup: Initializing a Flask application to handle HTTP requests and responses.
- Request Handling: Creating endpoints to process user inputs, tokenize them, and retrieve the appropriate first aid responses from the JSON dataset.

- Algorithm Integration: Implementing the tokenization and matching algorithms, including the Levenshtein distance calculation, to find the most relevant responses.
- Response Generation: Formatting and sending the retrieved first aid instructions back to the frontend for display to the user.

4.5.2 JSON INTEGRATION

The backend server integrates with a structured JSON file that contains mappings of symptoms to first aid responses. This integration involves following.

- Data Loading: Loading the JSON dataset into the server memory for quick access.
- Data Access: Implementing functions to search and retrieve data from the JSON file based on user input tokens.
- Performance Optimization: Ensuring that data retrieval is efficient and can handle multiple user requests in real-time.

The structured JSON file is designed to facilitate rapid searching and retrieval, ensuring that users receive timely and accurate first aid instructions.

4.5.3 FRONTEND DEVELOPMENT

The frontend is developed using HTML and CSS to create an intuitive and user-friendly interface.

- User Interface Design: Designing a clean and responsive interface where users can input their symptoms and receive real-time first aid suggestions.
- Form Handling: Creating input forms to capture user symptoms and send them to the backend server for processing.
- Real-Time Interaction: Implementing asynchronous communication (using AJAX or Fetch API) to interact with the backend server without reloading the page.
- Response Display: Formatting and displaying the first aid instructions received from the backend in a clear and actionable manner.

By combining HTML and CSS, the frontend provides a visually appealing and functional interface that enhances user experience and ensures easy interaction with the chatbot. The implementation phase brings together the backend, JSON dataset, and frontend to create a comprehensive and efficient First Aid Chatbot. Python Flask is used to develop the backend server, ensuring robust request handling and data retrieval.

CHAPTER 5

SYSTEM ARCHITECTURE

5.1 ARCHITECTURE DIAGRAM

Figure 5.1 illustrates the overall architecture of the First Aid Chatbot system. Included Components are given below.

- Frontend Interface (HTML/CSS): Provides a user-friendly interface for users to input symptoms and receive first aid suggestions.
- Backend Server (Python Flask): Processes user inputs, tokenizes them, and searches the JSON dataset for matching symptoms.
- JSON Dataset: Contains structured mappings of symptoms to corresponding first aid responses.
- Tokenization Algorithm: Converts user inputs into tokens and facilitates efficient search and retrieval from the JSON dataset.

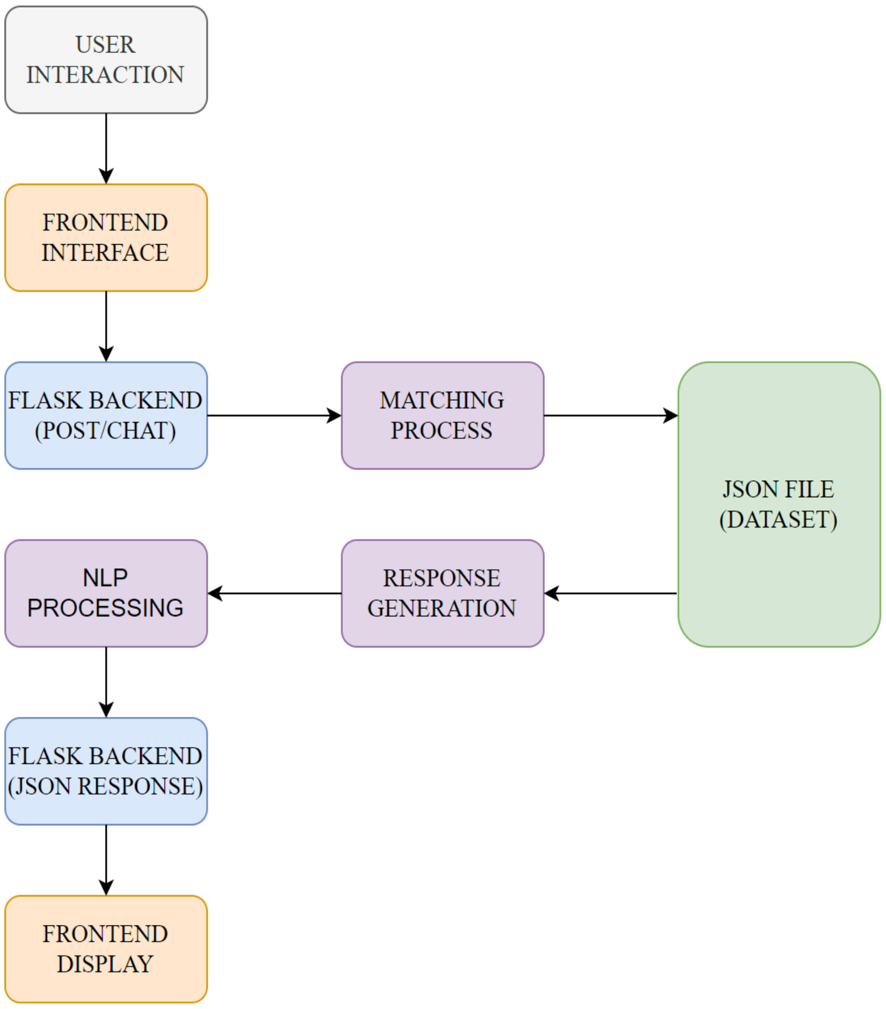


Figure 5.1 ARCHITECTURE DIAGRAM

Initially, users interact with the system via a frontend interface, which captures their input and sends it to a Flask backend using a POST request. The backend initiates the matching process, querying a JSON file dataset to find relevant information. This matching process involves searching the dataset for the most relevant entries based on the user's input. The results from this matching process are then fed into a response generation module, which formulates appropriate responses [4]. These responses undergo NLP processing to ensure they are coherent, contextually appropriate, and tailored to the user's input. Once the NLP processing is complete, the Flask backend sends the generated response back to the frontend in a JSON format. The frontend then displays the response to the user, completing the interaction cycle. This architecture ensures a seamless flow of information, from user input to response generation, leveraging both backend processing and NLP capabilities.

5.2 WORKFLOW

Figure 5.2 outlines the Working of handling user interactions related to First Aid Queries.

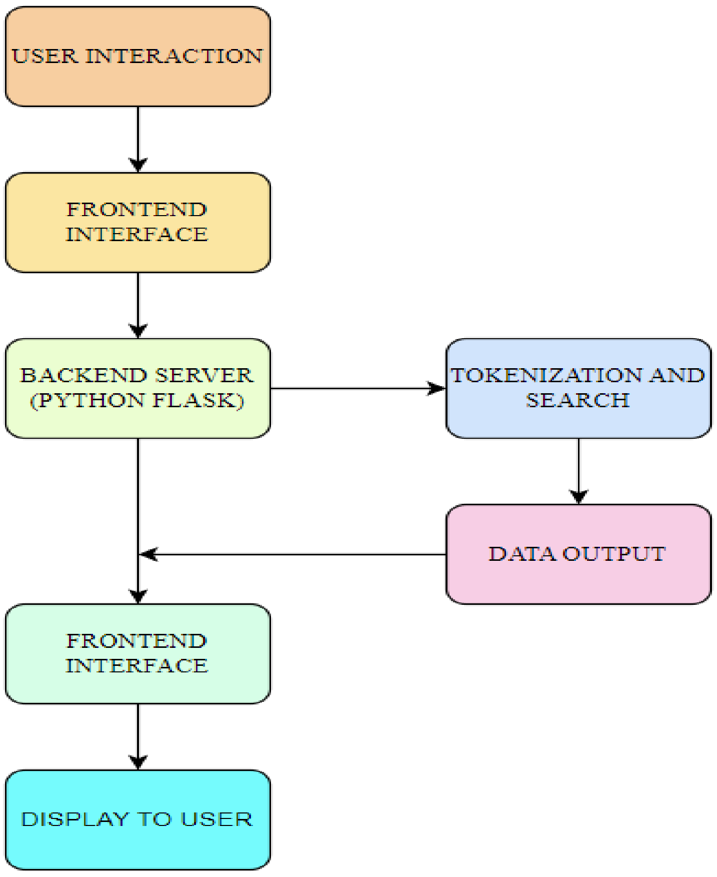


Figure 5.2 WORKFLOW DIAGRAM

Users begin by entering symptoms or questions into the frontend interface. This input is then sent to the backend server, built with Python Flask, via HTTP requests. The backend processes the input by invoking a tokenization algorithm, which breaks down the input into tokens. These tokens facilitate efficient searches within a JSON dataset to retrieve relevant first aid responses. The backend then formats these responses and sends them back to the frontend interface. Finally, the frontend displays the formatted first aid instructions in a user-friendly manner, ensuring users receive clear and actionable guidance based on their queries.

- User Interaction: Users engage with the frontend interface, entering symptoms or queries related to first aid situations. This could involve typing in symptoms they are experiencing or asking questions about specific first aid procedures.
- Backend Processing: The frontend interface sends these user inputs to the backend server, built with Python Flask, via HTTP requests. Upon receiving these requests, the Flask backend processes them by invoking a tokenization algorithm. This algorithm breaks down the user inputs into smaller units called tokens, which are easier to work with for search operations.
- Tokenization and Search: The tokenization algorithm parses the user inputs, converting them into tokens. These tokens are then used to perform efficient search operations within the JSON dataset. The backend searches the dataset using the tokenized symptoms or queries as keys, retrieving the most relevant first aid responses.
- Data Output: The retrieved first aid instructions are formatted appropriately and sent back to the frontend. The frontend then renders these responses in a user-friendly format, ensuring that the users receive clear and actionable first aid guidance based on their input. This seamless interaction provides users with immediate assistance and relevant first aid information.

5.3 LEVENSHTEIN ALGORITHM

The Levenshtein algorithm also called Levenshtein edit distance, which means the number of differences between two words called like distance. This variable is also used as a parameter to check how much difference can be tolerated. The Levenshtein distance between two words is the smallest number of single-character modifications (insertions, deletions, or substitutions) required to transform one word into the other. It is named after Vladimir Levenshtein, a Soviet mathematician who studied this distance in 1965. Dynamic Programming requires that you first be able to solve similar problems, to apply the technique to the particular problem you are trying to solve. Therefore, in this algorithm, we have divided the problem into 2 steps to figure out the distinctions and find a solution way according to this map.

5.3.1 LEVENSHTEIN MATRIX

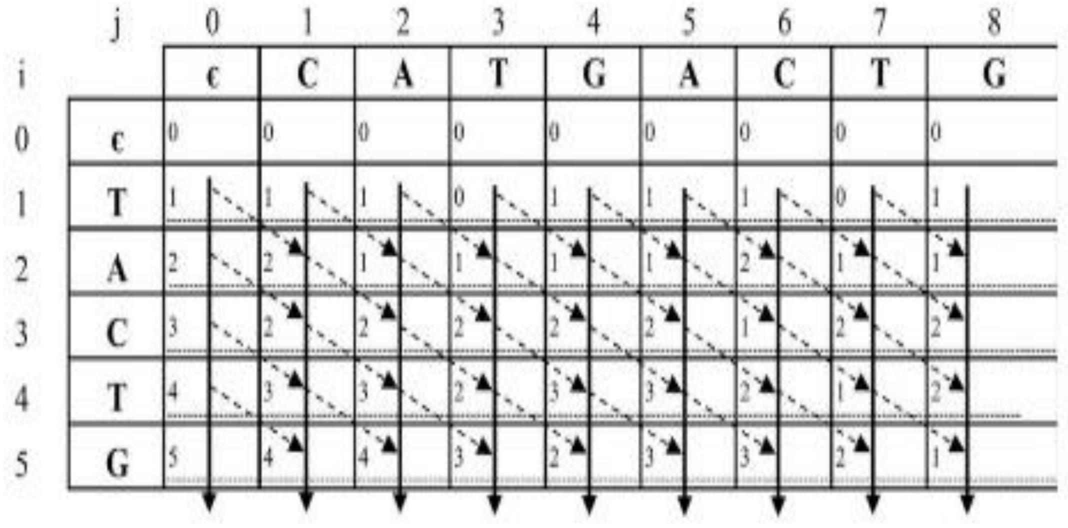
The Process of Levenshtein consists of two parts, which are forming the matrix by crosschecking the letters of words and giving the value for each cell according to the logic of the algorithm and the backtracking technique to announce which operation has to be done to fix a word at the end optimally.

Table 5.1 LEVENSHTEIN MATRIX

		E	L	E	P	H	A	N	T
	0	1	2	3	4	5	6	7	8
R	1	1	2	3	4	5	6	7	8
E	2	1	2	2	3	4	5	6	7
L	3	2	1	2	3	4	5	6	7
E	4	3	2	1	2	3	4	5	6
V	5	4	3	2	2	3	4	5	6
A	6	5	4	3	3	3	3	4	5
N	7	6	5	4	4	4	4	3	4
T	8	7	6	5	5	5	5	4	3

In the first step, as we can see from Table 5.1, we have to create a matrix in that words are placed on the rows and columns no matter whether the number of the letters is the same or different. Firstly, we place the initial values on words according to their order from beginning to end. After every letter, we increment the value by one. In both words cells next to them are filled with values in ascending order naturally. As we can see from Figure 1, the “RELEVANT” word has values for every letter next to them in ascending order from 1 to 8 since it is 8 letters. Besides, that same condition is valid for the other word “ELEPHANT” shown in the figure as well. Therefore, there is an extra cell that is assigned as 0 because there is no letter in a column or row naturally. At the beginning of the two words comparison start and goes on at the end of the matrix the last element on the diagonal. In this manner, comparison can be approached cell by cell and the responding row and column letter to that cell. In the beginning, the trivial cell which has 0 value on the leftmost and the top one is selected, and there is no letter for his cell. When we pass to the next cell one by one, we need to imply some execution to make the right action according to the algorithm.

For every cell, if the compared letters are equal then we have to assign the current cell's previous diagonal cell value directly. Otherwise, if the compared letters are not the same as each other, then it increments the three values around it at the left, top, and diagonal upperleft. After incrementing, the smallest value of those results is selected as the new value of the present cell. These implementations are applied to all empty cells from beginning to end cells gradually as shown in Figure 5.3.



i \ j		0	1	2	3	4	5	6	7	8
		ϵ	C	A	T	G	A	C	T	G
0	ϵ	0	0	0	0	0	0	0	0	0
1	T	1	1	1	0	1	1	1	0	1
2	A	2	2	1	1	1	1	2	1	1
3	C	3	2	2	2	2	2	1	2	2
4	T	4	3	3	2	3	3	2	1	2
5	G	5	4	4	3	2	3	3	2	1

Figure 5.3 LEVENSHTEIN MATRIX - 2

The Levenshtein algorithm (also called Livan-Distance) calculates the lowest number of editing processes necessary to modify one series to get another series. The most common way to calculate this is by the dynamic programming approach. A matrix is initialized to Each cell always tries to minimize the cost locally as shown in the algorithm in Equation below. The second step is deciding which operations need to be executed to make both strings in the same form.

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j)=0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise} \end{cases} \quad \text{-------------Eq. (5-1)}$$

For this process, we need to complete the Levenshtein matrix based on the procedure explained above. In this matrix, we have to focus on the last element of the matrix that is located at the rightmost and lowest cell in the whole grid. For example, if the length of the words is n and m then this first selected cell can be said that the mth and nth cell in the matrix. Each cell always tries to minimize the cost locally as shown in the algorithm in Eq. (5-1).

5.3.2 OPERATION DECISION LOGIC

Operation Decision Logic is shown in Figure 5.4. After this first cell, we control the three cell around the current cell which is upper, left, and left upper (diagonal) and the minimum of three cell values is selected as the target cell and our cursor move there. Before we move there, a decision has to be made about which operation needs to be done. In addition to that, if replacement is selected as the optimal operation, we need to decide which letters require exchanging each other.

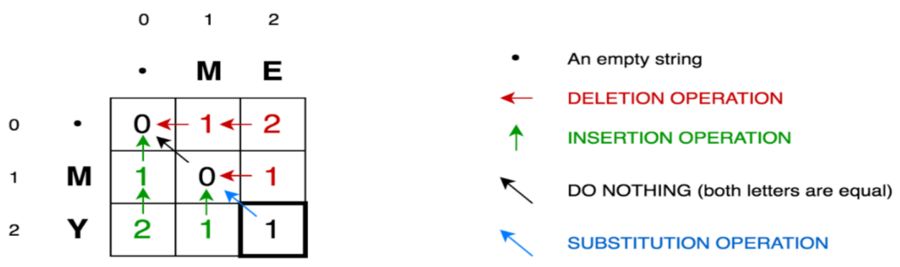
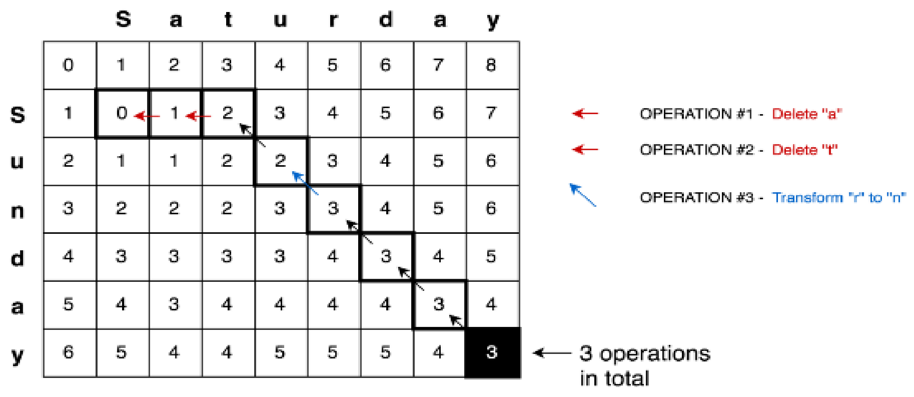


Figure 5.4 OPERATION DECISION LOGIC

If the minimum value is in the left cell, we can conclude that our operation should be arranged as the deleting current letter that is controlled against others. If the minimum value is in an upper cell, we can deduce that the needed operation for the appropriate solution is insertion. Finally, if the diagonal cell has the least value, then this means a replacement operation should be included in the list of required operations. If no minimum value is found, i.e, all cells are equal, then we skip this point without doing anything because they are the same letters exactly.

5.3.3 BACKTRACKING



OPERATION #1 - Delete "a"
OPERATION #2 - Delete "t"
OPERATION #3 - Transform "r" to "n"
3 operations in total

Figure 5.5 BACKTRACKING

As we can see from Figure 5.5, until the first element is reached, the backtracking operation is going on. When backtracking is implemented, an extra variable is held to record how many operations are done in a way that after every operation this value is incremented by 1. Generally, the targeted word is settled at the row of the compromised matrix. For deletion and insertion, letters are chosen from its row index for the desired word on that side.

5.3.4 PSEUDOCODE FOR LEVENSHTEIN RECURSION

The Levenshtein algorithm has an effective result for the area of string comparison that diversifies as DNA line-up researches, search engines recommendation even though in the link prediction, cryptography and recognition of the images in the machine learning recently. Besides that, some tests compared the performance of approximate subgraph matching with the string edit distance approach. This approach outperformed the approximate subgraph matching method in terms of computational cost and accuracy. When the sort of the string letters is changed, it affects the result also, as all values need to be recalculated. The Levenshtein distance is the most widely used of the edit distance family of distance metrics. The collection of elementary operations allowed to conduct the transformation differs between these sibling distance metrics. For example, Hamming distance only allows substitutions. There are various kinds of Levenshtein methods apart from what we discussed. One of the Levenshtein methods is a straightforward recursive algorithm which is shown below.

```
Function LEVDIST (a, b) is
    if a.size() == 0 then
        return b.size();
    end
    if b.size() == 0 then
        return a.size();
    end
    indicator ← a [0] ≠ b [0]? 1: 0;
    return Minimum (
        LEVDIST(a.substr(1), b) + 1,          //deletion
        LEVDIST(b.substr(1), a) + 1,          // insertion
        LEVDIST(a.substr(1), b.substr(1)) + indicator
            //substitution
    );
end
```

5.3.5 FULL MATRIX VERSION OF LEVENSHTEIN

The function x.substr(n) returns a substring of x beginning at element n. This approach is inefficient because it calculates distance for the same prefixes many times. Another version is iterative with a full matrix; knowing the features of the Levenshtein distance, we can see that a matrix of dimension (|a| + 1) times (|b| +1] can be constructed, including the value lev a,b I j) at the point I j. The first row and the first column of this matrix are set to the values in the ranges 0..a and 0..b, respectively.

```
Function LEVDIST (a, b) is
    n ← a.size();
    m ← b.size();
    distance [i, 0] ← i ∀ i ∈ 0.n;
    distance [0, j] ← j ∀ j ∈ 0.m;
    for j ← 1 to m do
        for i ← 1 to n do
            indicator ← a[i] ≠ b[j]? 1: 0;
            distance [i, j] ← Minimum (
            distance [i − 1, j] + 1,    // deletion
            distance [i, j − 1] + 1,     // insertion
            distance [i − 1, j − 1] + indicator // substitution
            );
        end
    end
    return distance [n, m];
end
```

We can use a dynamic programming approach to fill in this matrix to obtain the final, bottom-right element as our resulting distance. When we implement this version, we will get the how many operations to need to be done directly at the end of the matrix as below example.

Table 5.2 COMPARISION MATRIX

		E	**X**	**E**	**C**	**U**	**T**	**I**	**O**	**N**
	0	1	2	3	4	5	6	7	8	9
I	1	1	2	3	4	5	6	6	7	8
N	2	2	2	3	4	5	6	7	7	7
T	3	3	3	3	4	5	5	6	7	8
E	4	3	4	3	4	5	6	6	7	8
N	5	4	4	4	4	5	6	7	7	7
T	6	5	5	5	5	5	5	6	7	8
I	7	6	6	6	6	6	6	5	6	7
O	8	7	7	7	7	7	7	6	5	6
N	9	8	8	8	8	8	8	7	6	5

Table 5.2 is the Comparision Matrix. 5 is the result of this comparison and every operation is counted as 1 for the full matrix as well. The bottom-right element of this matrix is the same as the five operations we observed previously. Another version is iterative with two rows. If we want to gain the final value alone, we can easily modify the implementation of the above-mentioned provisions to avoid the allocation of the entire matrix. To move forward, we only need two rows - the one we are currently updating and the previous one. Below code illustrates Two Rows Version. This optimization makes it impossible to determine which edits were made. Hirschberg's algorithm solves this problem using both dynamic programming and division and conquer.

```
Function LEVDIST(a, b) is
    n ← a.size();
    m ← b.size();
    previous[j] ← j ∀ j ∈ 0..m;
    for i ← 0 to n − 1 do
        current[0] = i + 1;
        for j ← 0 to m − 1 do
            indicator ← a[i] ≠ b[j] ? 1 : 0;
            current[j + 1] ← Minimum(
                previous[j + 1] + 1,      // deletion
                current[j] + 1,           // insertion
                previous[j] + indicator   // substitution
            );
        end
        previous ← current;   /* copy current row to previous row
                                 for next iteration */
    end
    return previous[m];   /* result is in the previous row after the last
                             swap */
end
```

5.3.6 LEV DISTANCE APPROACH

Furthermore, we may observe the fact that to calculate the value at the specific row position we need only three values – the one to the left, the one directly above, and the last one diagonal. Lev Distance Approach is shown below.

```
Function LEVDIST(a, b) is
    n ← a.size();
    m ← b.size();
    row[j] ← j ∀ j ∈ 0..m;
        for i ← 0 to n − 1 do
            row[0] = i + 1;
            for j ← 0 to m − 1 do
                indicator ← a[i] ≠ b[j] ? 1 : 0;
                previousDiagonal ← previousAbove;
                previousAbove ← row[j + 1];
                row[j + 1] ← Minimum(
              previousAbove + 1,       // deletion
                row[j] + 1,            // insertion
              previousDiagonal + indicator
                                // substitution
              );
            end
        end
        return row[m];
end
```

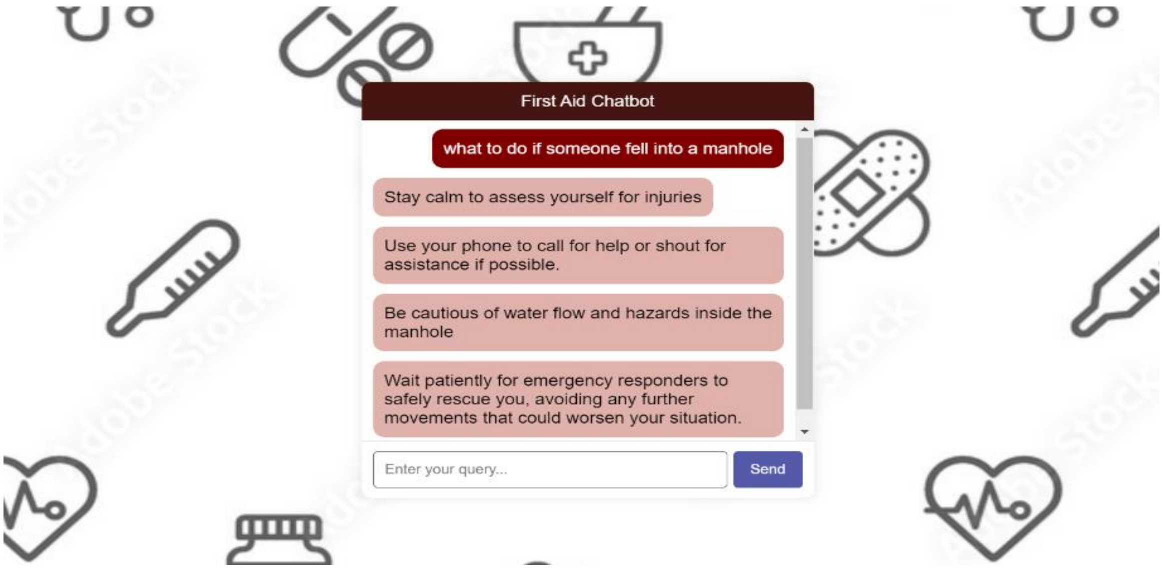
When we consider this algorithm, according to complexity, we can say that the length of the words is the main parameter. The time complexity of all the iterative algorithms presented above is O(|a| x |b|). Space complexity for the full matrix implementation is O(|a| x |b|) which usually makes it impractical to use. Both two-rows and single-row implementations provide linear space complexity O(max(|a| , |b|)) . Swapping source and target to reduce computation row length will further reduce it to O(min(|a|, |b|)). It has been shown that the Levenshtein distance cannot be calculated in subquadratic time unless the strong exponential time hypothesis is false. Fortunately, this is only a partial description of the complexity of the problem.

When we take this algorithm in terms of upper boundary and minimum distance, we can say that some combined methods are used. Let's say we have a large string and want to compare only similar strings, such as misspelled names. Complete Levenshtein computation would have to traverse the full matrix in this scenario, including the high values in the topright and bottom-left corners that we won't require. This gives us an idea of how the threshold could be improved, with all distances above a certain boundary simply being reported as out of range. As a result, we only need to compute the values in the diagonal stripe of width 2K + 1 for bounded distance, where K is the distance threshold. In other words, if the Levenshtein distance exceeds the boundary, the implementation will fail. This method provides us with the time complexity of O(min(|a|,|b|)), which allows us to execute large but comparable strings in a reasonable amount of time. We can also skip the calculation if the distance exceeds the threshold we set because we know the distance is at least the length difference between the strings.

CHAPTER 6

RESULTS

6.1 CHATBOT INTERFACE



what to do if someone fell into a manhole

Stay calm to assess yourself for injuries

Use your phone to call for help or shout for assistance if possible.

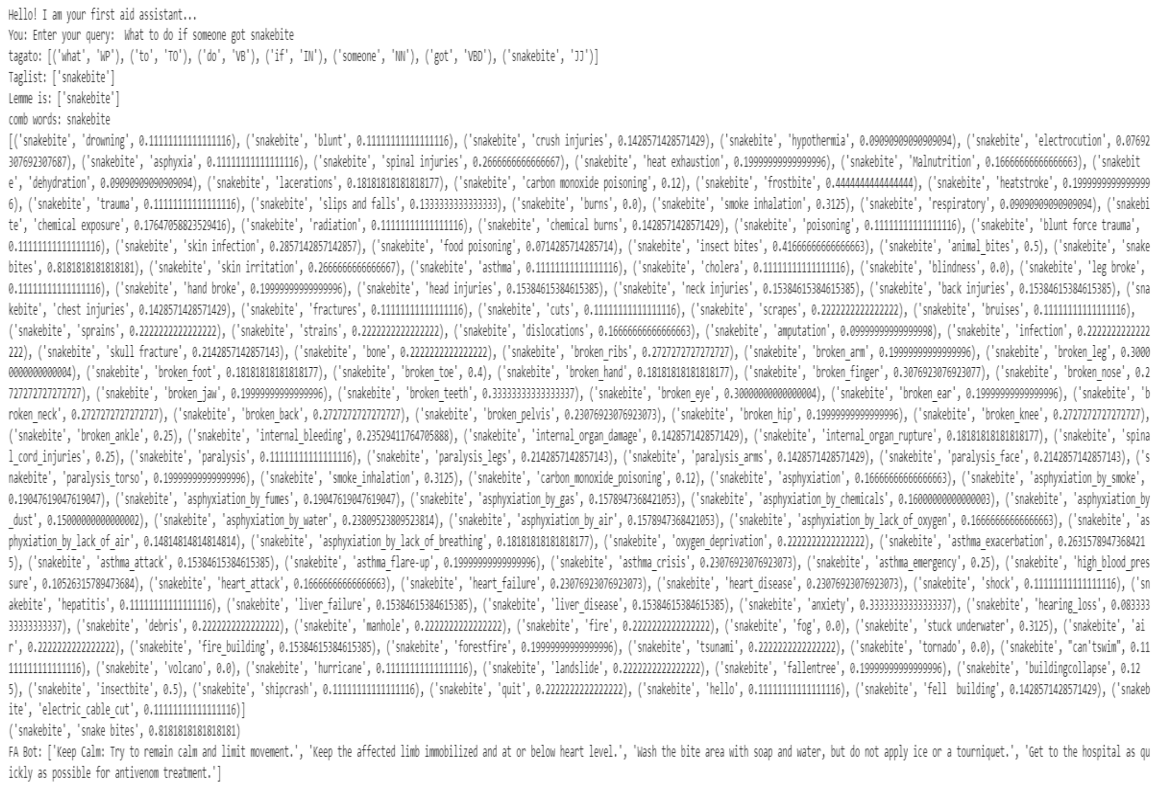
Be cautious of water flow and hazards inside the manhole

Wait patiently for emergency responders to safely rescue you, avoiding any further movements that could worsen your situation.

Figure 6.1 CHATBOT INTERFACE

In the Above Figure 6.1, It shows the Chatbot Interface for Users where User can Input Queries Related to First Aid.

6.2 INTERNAL WORKING OF CHATBOT



```
Hello! I am your first aid assistant...
You: Enter your query:  What to do if someone got snakebite
tagato: [('what', 'WP'), ('to', 'TO'), ('do', 'VB'), ('if', 'IN'), ('someone', 'NN'), ('got', 'VBD'), ('snakebite', 'JJ')]
Taglist: ['snakebite']
Lemme is: ['snakebite']
comb words: snakebite
[('snakebite', 'drowning', 0.11111111111111116), ('snakebite', 'blunt', 0.11111111111111116), ('snakebite', 'crush injuries', 0.1428571428571429), ('snakebite', 'hypothermia', 0.09090909090909094), ('snakebite', 'electrocution', 0.07692
307692307687), ('snakebite', 'asphyxia', 0.11111111111111116), ('snakebite', 'spinal injuries', 0.2666666666666667), ('snakebite', 'heat exhaustion', 0.19999999999999996), ('snakebite', 'Malnutrition', 0.16666666666666663), ('snakebit
e', 'dehydration', 0.09090909090909094), ('snakebite', 'lacerations', 0.18181818181818177), ('snakebite', 'carbon monoxide poisoning', 0.12), ('snakebite', 'frostbite', 0.4444444444444444), ('snakebite', 'heatstroke', 0.1999999999999999
6), ('snakebite', 'trauma', 0.11111111111111116), ('snakebite', 'slips and falls', 0.1333333333333333), ('snakebite', 'burns', 0.0), ('snakebite', 'smoke inhalation', 0.3125), ('snakebite', 'respiratory', 0.09090909090909094), ('snakebi
te', 'chemical exposure', 0.17647058823529416), ('snakebite', 'radiation', 0.11111111111111116), ('snakebite', 'chemical burns', 0.1428571428571429), ('snakebite', 'poisoning', 0.11111111111111116), ('snakebite', 'blunt force trauma',
0.11111111111111116), ('snakebite', 'skin infection', 0.2857142857142857), ('snakebite', 'food poisoning', 0.0714285714285714), ('snakebite', 'insect bites', 0.41666666666666663), ('snakebite', 'animal_bites', 0.5), ('snakebite', 'snake
bites', 0.8181818181818181), ('snakebite', 'skin irritation', 0.2666666666666667), ('snakebite', 'asthma', 0.11111111111111116), ('snakebite', 'cholera', 0.11111111111111116), ('snakebite', 'blindness', 0.0), ('snakebite', 'leg broke',
0.11111111111111116), ('snakebite', 'hand broke', 0.19999999999999996), ('snakebite', 'head injuries', 0.15384615384615385), ('snakebite', 'neck injuries', 0.15384615384615385), ('snakebite', 'back injuries', 0.15384615384615385), ('sna
kebite', 'chest injuries', 0.1428571428571429), ('snakebite', 'fractures', 0.11111111111111116), ('snakebite', 'cuts', 0.11111111111111116), ('snakebite', 'scrapes', 0.2222222222222222), ('snakebite', 'bruises', 0.11111111111111116),
('snakebite', 'sprains', 0.2222222222222222), ('snakebite', 'strains', 0.2222222222222222), ('snakebite', 'dislocations', 0.16666666666666663), ('snakebite', 'amputation', 0.09999999999999998), ('snakebite', 'infection', 0.222222222222
222), ('snakebite', 'skull fracture', 0.2142857142857143), ('snakebite', 'bone', 0.2222222222222222), ('snakebite', 'broken_ribs', 0.2727272727272727), ('snakebite', 'broken_arm', 0.19999999999999996), ('snakebite', 'broken_leg', 0.3000
0000000000004), ('snakebite', 'broken_foot', 0.18181818181818177), ('snakebite', 'broken_toe', 0.4), ('snakebite', 'broken_hand', 0.18181818181818177), ('snakebite', 'broken_finger', 0.3076923076923077), ('snakebite', 'broken_nose', 0.2
727272727272727), ('snakebite', 'broken_jaw', 0.19999999999999996), ('snakebite', 'broken_teeth', 0.33333333333333337), ('snakebite', 'broken_eye', 0.30000000000000004), ('snakebite', 'broken_ear', 0.19999999999999996), ('snakebite', 'b
roken_neck', 0.2727272727272727), ('snakebite', 'broken_back', 0.2727272727272727), ('snakebite', 'broken_pelvis', 0.23076923076923073), ('snakebite', 'broken_hip', 0.19999999999999996), ('snakebite', 'broken_knee', 0.2727272727272727),
('snakebite', 'broken_ankle', 0.25), ('snakebite', 'internal_bleeding', 0.23529411764705888), ('snakebite', 'internal_organ_damage', 0.1428571428571429), ('snakebite', 'internal_organ_rupture', 0.18181818181818177), ('snakebite', 'spina
l_cord_injuries', 0.25), ('snakebite', 'paralysis', 0.11111111111111116), ('snakebite', 'paralysis_legs', 0.2142857142857143), ('snakebite', 'paralysis_arms', 0.1428571428571429), ('snakebite', 'paralysis_face', 0.2142857142857143), ('s
nakebite', 'paralysis_torso', 0.19999999999999996), ('snakebite', 'smoke_inhalation', 0.3125), ('snakebite', 'carbon_monoxide_poisoning', 0.12), ('snakebite', 'asphyxiation', 0.16666666666666663), ('snakebite', 'asphyxiation_by_smoke',
0.19047619047619047), ('snakebite', 'asphyxiation_by_fumes', 0.19047619047619047), ('snakebite', 'asphyxiation_by_gas', 0.1578947368421053), ('snakebite', 'asphyxiation_by_chemicals', 0.16000000000000003), ('snakebite', 'asphyxiation_by
_dust', 0.15000000000000002), ('snakebite', 'asphyxiation_by_water', 0.23809523809523814), ('snakebite', 'asphyxiation_by_air', 0.1578947368421053), ('snakebite', 'asphyxiation_by_lack_of_oxygen', 0.16666666666666663), ('snakebite', 'as
phyxiation_by_lack_of_air', 0.14814814814814814), ('snakebite', 'asphyxiation_by_lack_of_breathing', 0.18181818181818177), ('snakebite', 'oxygen_deprivation', 0.2222222222222222), ('snakebite', 'asthma_exacerbation', 0.2631578947368421
5), ('snakebite', 'asthma_attack', 0.15384615384615385), ('snakebite', 'asthma_flare-up', 0.19999999999999996), ('snakebite', 'asthma_crisis', 0.23076923076923073), ('snakebite', 'asthma_emergency', 0.25), ('snakebite', 'high_blood_pres
sure', 0.10526315789473684), ('snakebite', 'heart_attack', 0.16666666666666663), ('snakebite', 'heart_failure', 0.23076923076923073), ('snakebite', 'heart_disease', 0.23076923076923073), ('snakebite', 'shock', 0.11111111111111116), ('sn
akebite', 'hepatitis', 0.11111111111111116), ('snakebite', 'liver_failure', 0.15384615384615385), ('snakebite', 'liver_disease', 0.15384615384615385), ('snakebite', 'anxiety', 0.33333333333333337), ('snakebite', 'hearing_loss', 0.083333
33333333337), ('snakebite', 'debris', 0.2222222222222222), ('snakebite', 'manhole', 0.2222222222222222), ('snakebite', 'fire', 0.2222222222222222), ('snakebite', 'fog', 0.0), ('snakebite', 'stuck underwater', 0.3125), ('snakebite', 'ai
r', 0.2222222222222222), ('snakebite', 'fire_building', 0.15384615384615385), ('snakebite', 'forestfire', 0.19999999999999996), ('snakebite', 'tsunami', 0.2222222222222222), ('snakebite', 'tornado', 0.0), ('snakebite', "can'tswim", 0.11
111111111111116), ('snakebite', 'volcano', 0.0), ('snakebite', 'hurricane', 0.11111111111111116), ('snakebite', 'landslide', 0.2222222222222222), ('snakebite', 'fallentree', 0.19999999999999996), ('snakebite', 'buildingcollapse', 0.12
5), ('snakebite', 'insectbite', 0.5), ('snakebite', 'shipcrash', 0.11111111111111116), ('snakebite', 'quit', 0.2222222222222222), ('snakebite', 'hello', 0.11111111111111116), ('snakebite', 'fell  building', 0.1428571428571429), ('snakeb
ite', 'electric_cable_cut', 0.11111111111111116)]
('snakebite', 'snake bites', 0.8181818181818181)
FA Bot: ['Keep Calm: Try to remain calm and limit movement.', 'Keep the affected limb immobilized and at or below heart level.', 'Wash the bite area with soap and water, but do not apply ice or a tourniquet.', 'Get to the hospital as qu
ickly as possible for antivenom treatment.']
```

Figure 6.2 INTERNAL WORKING OF CHATBOT

Figure 6.2 Shows how the Chatbot is Working Internally using Levenshtein Algorithm.

6.3 INPUT TESTING

The Table 6.1 presents a set of user queries related to various emergency situations along with their corresponding key tags and whether the system produces a response. The table is designed to categorize different types of emergencies, such as fire, skin problems, landslides, snake bites, drowning, and building collapses. For each query, the key tag represents the main subject of the query, and the "Yes" in the output/response production column indicates that the system is capable of providing a relevant response or advice for each situation. This structure helps in efficiently managing and responding to diverse emergency scenarios.

Table 6.1 INPUT TESTING

USER INPUT / QUERY	KEY / TAG	OUTPUT / RESPONSE PRODUCTION
Where To Go When There is Fire in Building?	Fire	Yes
How to cure skin allergy?	Skin Problems	Yes
Where to go when there is a Landslide?	Landslide	Yes
I Got Bit by a Snake	Snake Bite	Yes
What to do if someone is Drowning?	Drowning	Yes
Building is Collapsed	Building Collapse	Yes

CHAPTER 7

CONCLUSION AND FUTURE WORK

The First Aid Chatbot represents a transformative approach to delivering essential medical guidance during emergencies. By harnessing the power of advanced technologies such as natural language processing (NLP) and machine learning, the chatbot can provide precise and timely first aid advice tailored to various disaster scenarios. The integration of real-time data ensures that the information remains current and accurate, while support for multiple languages guarantees accessibility for diverse user populations. In high-stress and chaotic disaster situations, the chatbot's user-friendly interface enables individuals to quickly and easily obtain critical first aid information, potentially saving lives and reducing the severity of injuries. The system's continuous improvement through user interaction data further enhances its effectiveness and reliability over time. The First Aid Chatbot is a vital tool that significantly enhances disaster response efforts. Its ability to provide immediate, accurate, and accessible first aid information makes it an invaluable resource for individuals facing emergencies, ultimately contributing to better health outcomes and increased resilience in the face of disasters.

Expanding your first aid chatbot project has significant potential for future development. Integration with medical devices, such as wearable technology and smart home systems, can enhance real-time monitoring and immediate assistance capabilities. Advancing the chatbot's AI and machine learning capabilities will enable personalized advice based on individual health profiles and improved natural language processing for handling complex medical queries. Additionally, expanding multi-language support will make the chatbot accessible to a broader audience, providing accurate and culturally relevant advice. Integration with emergency services, including automated alerts to emergency contacts or services, can ensure timely assistance in critical situations.

Furthermore, there are gaps not yet covered those present opportunities for enhancement. These include incorporating visual recognition to diagnose injuries or conditions from images or videos, and expanding the database to include more comprehensive and up-to-date medical information. Additionally, integrating with telemedicine services can allow users to connect with healthcare professionals directly through the chatbot, providing an extra layer of support and advice.

REFERENCES

[1] S. Anushka and S. Thelijjagoda, "Healbot: NLP-based Health Care Assistant for Global Pandemics," 2022 International Research Conference on Smart Computing and Systems Engineering (SCSE), Colombo, Sri Lanka, pp. 61-67, doi:10.1109/SCSE56529.2022.9905107.

[2] Sreedhar Kumar S, Syed Thouheed Ahmed, Afifa Salsabil Fathima, Nishabai M, Sophia S, “Medical ChatBot Assistance for Primary Clinical Guidance using Machine Learning Techniques”, 2024, https://doi.org/10.1016/j.procs.2024.03.217.

[3] K. Arumugam, Mohd Naved, Priyanka P. Shinde, Orlando Leiva-Chauca, Antonio Huaman-Osorio, Tatiana Gonzales-Yanac, “Multiple disease prediction using Machine learning algorithms”, 2023, https://doi.org/10.1016/j.matpr.2021.07.361.

[4] Yang, Christopher C, “Explainable Artificial Intelligence for Predictive Modeling in Healthcare”, 2022 Journal of Healthcare Informatics Research, https://doi.org/10.1007/s41666-022-00114-1.

[5] Shashidhar, R. Patilkulkarni, S., Puneeth, S. B., “Combining audio and visual speech recognition using LSTM and deep convolutional neural network”, 2022 International Journal of Information Technology, https://doi.org/10.1007/s41870-022-00907-y.

[6] Chou, J. S., Chong, P. L., & Liu, C. Y. “Deep learning-based chatbot by natural language processing for supportive risk management in river dredging projects”. Engineering Applications of Artificial Intelligence, 131, 107744, 2024. https://doi.org/10.1016/j.engappai.2023.107744.

[7] Ouerhani, N., Maalel, A. & Ben Ghézela, H. “SPeCECA: a smart pervasive chatbot for emergency case assistance based on cloud computing”. Cluster Comput 23, 2471–2482, 2020. https://doi.org/10.1007/s10586-019-03020-1.

[8] Ayanouz, S., Abdelhakim, B. A., & Benhmed, M. (2020, March). “A smart chatbot architecture based NLP and machine learning for health care assistance”. In Proceedings of the 3rd international conference on networking, information systems & security (pp. 1-6). https://doi.org/10.1145/3386723.3387897.

[9] Soufyane, A., Abdelhakim, B. A., & Ahmed, M. B. (2021, January). “An intelligent chatbot using NLP and TF-IDF algorithm for text understanding applied to the medical field”. In Emerging Trends in ICT for Sustainable Development: The Proceedings of NICE2020 International Conference (pp. 3-10). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-53440-0_1.

[10] Mokmin, N. A. M., & Ibrahim, N. A. (2021). “The evaluation of chatbot as a tool for health literacy education among undergraduate students”. Education and

Information Technologies, 26(5), 6033-6049. https://doi.org/10.1007/s10639-021-10542-y.

[11] Ouerhani, N., Maalel, A., & Ben Ghézala, H. (2022). “SMAD: SMart assistant during and after a medical emergency case based on deep learning sentiment analysis: The pandemic COVID-19 case”. Cluster computing, 25(5), 3671-3681. https://doi.org/10.1007/s10586-022-03601-7.

[12] Ryu, H., Kim, S., Kim, D., Han, S., Lee, K., & Kang, Y. (2020). “Simple and steady interactions win the healthy mentality: designing a chatbot service for the elderly”. Proceedings of the ACM on human-computer interaction, 4(CSCW2), 1-25. https://doi.org/10.1145/3415223.