iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

CHATBOT FOR MEDICAL ASSISTANCE USING MACHINE LEARNING

Guide: Mr. K. Suresh,
Gorreguttamidhi Srikanth, Gugulothu Lalitha, Gugulothu Siddhu, Kummari Vamshi

Department of Computer Science and Engineering, Joginpally B.R Engineering College
gorresrikanth06@gmail.com, lalithag2912@gmail.com, Siddhugugulothu03@gmail.com, kummarivamshi26@gmail.com

ABSTRACT
This project is a web-based Medical Chatbot designed to provide preliminary disease predictions based on user-provided symptoms. Built using the Flask framework, the application features an interactive conversational interface that allows patients to describe their ailments in everyday language. It also emphasizes security and data privacy by incorporating a user authentication system, which utilizes SQLite for session management and bcrypt for secure password hashing to protect personal user data.
At the core of the application is a machine-learning engine trained on a comprehensive dataset of symptoms and corresponding diseases. The system uses Natural Language Processing (NLP) techniques, specifically Term Frequency-Inverse Document Frequency (TF-IDF) vectorization, to mathematically represent and evaluate the significance of various symptoms. A Support Vector Machine (SVC) with a linear kernel was trained on this vectorized data, acting as a robust classifier that accurately maps combinations of symptoms to their most probable illnesses.
The data flow within the system begins when a logged-in user submits a message. A custom entity extraction module parses the conversational text, utilizing regular expressions to filter out non-medical terms and isolate specific recognized symptoms and diseases. Once the relevant medical entities are extracted, they are passed into the pre-trained SVC model for evaluation. In addition to predicting the condition, the system simultaneously queries an internal knowledge base to retrieve and suggest immediate medical precautions or treatments.
Overall, this application bridges the gap between early healthcare accessibility and artificial intelligence. By offering a fast, data-driven analysis of symptoms, it serves as an initial advisory tool to guide users toward the right health-conscious decisions and treatments. While it is not intended to replace professional medical diagnosis, it empowers users with quick insights and actionable health advice in a secure, user-friendly environment.

1. INTRODUCTION

In recent years, the rapid integration of artificial intelligence into the healthcare sector has significantly transformed how people access medical information. Despite the abundance of health resources available on the internet, individuals experiencing early symptoms often struggle to find accurate, personalized, and immediate guidance without physically visiting a healthcare professional. This gap in accessible, preliminary healthcare can lead to delayed diagnoses or unnecessary anxiety. To address this challenge, there is a growing need for intelligent, automated systems capable of analyzing symptoms and providing immediate, data-backed medical insights.
This project proposes a web-based Medical Chatbot, an interactive application designed to analyze user-provided symptoms and predict potential underlying diseases. By providing a conversational interface, the system allows users to intuitively describe how they are feeling in their own words. The backend processes this natural language input to extract key medical entities—filtering out irrelevant conversational filler—and cross-references the identified symptoms against a trained machine-learning classifier to generate a preliminary diagnosis. Furthermore, the application provides immediate suggestions for treatments and precautions, serving as a rapid, first-line advisory tool to help users decide on their next course of action.
The system is built using a modern, robust technology stack to ensure both accuracy and security. The core predictive engine utilizes a Support Vector Machine (SVC) algorithm, trained via the scikit-learn framework on a comprehensive dataset of symptoms and corresponding diseases. Natural Language Processing (NLP) techniques, such as TF-IDF vectorization, are employed to mathematically evaluate symptom relevance. Recognizing the highly sensitive nature of health data, the application also integrates a secure user authentication module—powered by SQLite and bcrypt password hashing—to ensure that all interactions remain confidential.
The primary objective of this project is not to replace professional medical consultations, but rather to bridge the gap between initial health concerns and clinical care.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Literature Survey

The application of Artificial Intelligence (AI) in preliminary healthcare and symptom analysis has evolved significantly, transitioning from static medical web portals to more dynamic and conversational diagnostic systems. Early symptom-checking applications primarily relied on rigid, rule-based decision trees where users were required to manually navigate complex menus and select symptoms from predefined dropdown lists. While these systems provided structured medical querying, they severely lacked natural language understanding, resulting in a frustrating user experience that failed to capture the nuanced ways patients describe their ailments. With the advancement of Natural Language Processing (NLP), conversational agents have been introduced to parse free-text medical queries. However, many of these generalized chatbots remain highly broad in scope, acting merely as information retrieval systems rather than leveraging dedicated mathematical models to accurately map symptoms to specific diseases.

In parallel, substantial research has been conducted in the domain of predictive medical modeling, leading to the development of highly accurate machine-learning classifiers using techniques such as Support Vector Machines (SVM), Naive Bayes, and decision trees. While these independent algorithms demonstrate strong performance in predicting diseases from symptom datasets, they are typically confined to backend research environments or standalone scripts. They largely lack integration with accessible, real-time conversational interfaces, preventing everyday users from directly benefiting from their predictive power. Furthermore, existing commercial telemedicine platforms and web-based symptom checkers (such as WebMD) often attempt to merge these features, yet they frequently suffer from bloated interfaces, lack an immediate real-time conversational flow, and often operate without dedicated, localized security protocols for persistent user session tracking.

These observations highlight a significant research gap: existing solutions frequently treat conversational NLP, rigorous machine-learning predictive modeling, and secure user management as isolated components rather than integrating them into a cohesive system. There is a notable lack of a unified, lightweight platform that enables intuitive natural language symptom reporting, real-time algorithmic diagnosis, and personalized medical precautions within a single web application. To address this gap, the proposed Medical Chatbot integrates NLP-driven entity extraction, a trained Support Vector Machine (SVC) diagnostic engine, and actionable treatment recommendations within a secure, Flask-based micro-architecture. By combining bcrypt-secured user authentication with TF-IDF vectorization and machine learning, this system provides a unified, interactive, and private ecosystem for preliminary medical exploration.

3. System Architecture:

A. Architecture Overview

The Medical Chatbot is designed using a robust client-server architecture to ensure secure request handling and low-latency responses. The web-based client interface interacts with the backend services through a centralized Flask API layer, while computationally intensive tasks such as Natural Language Processing (NLP) symptom extraction and machine learning predictions are handled by dedicated processing modules, as illustrated in Fig. 1.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

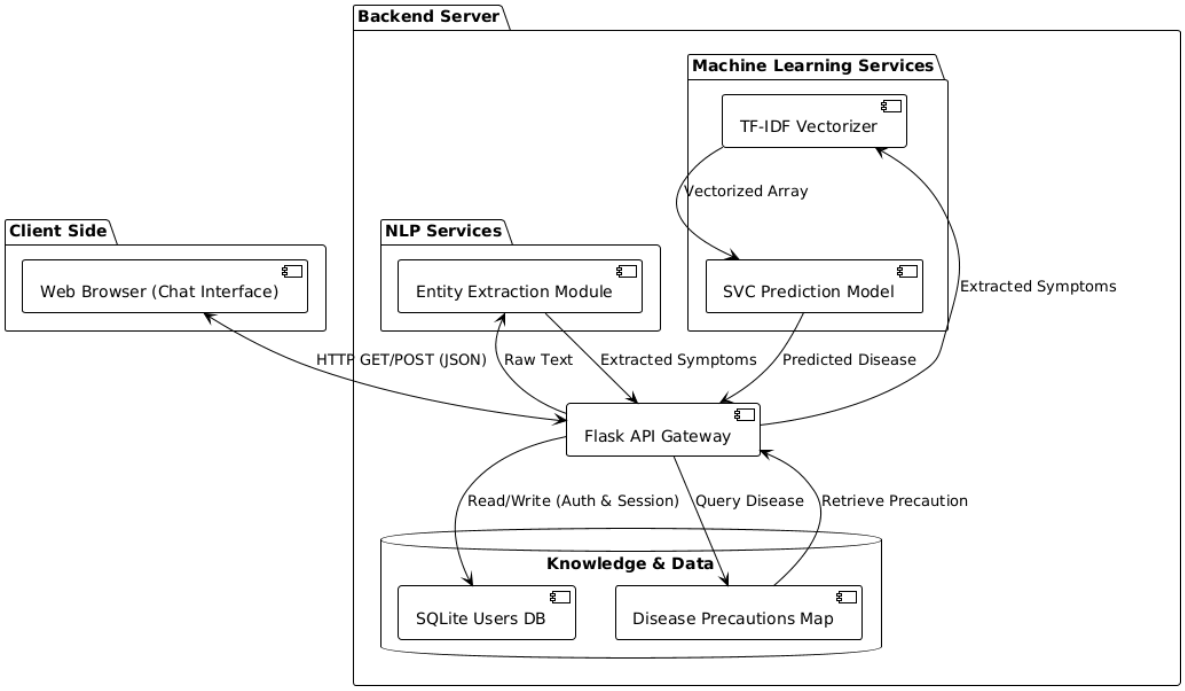


Fig. 1. *System Architecture*

B. **Core System Components:**

The Flask API Gateway acts as the central orchestration layer, handling request validation, secure password hashing (via bcrypt), and session management via SQLite database persistence. The NLP Extraction Module is a specialized service responsible for parsing raw conversational text, isolating valid medical symptoms, and filtering out non-medical conversational filler. The Predictive ML Engine processes the extracted symptoms to generate a disease diagnosis using a pre-trained Support Vector Machine (SVC). Before classification, the raw text is transformed into numerical data using a Term Frequency-Inverse Document Frequency (TF-IDF) mathematical model:

$$W(t,d) = TF(t,d) \times \log(N / DF(t))$$

where TF(t,d) represents the term frequency of symptom t in the user's query d, N is the total number of training documents in the dataset, and DF(t) is the number of documents containing the symptom term t.

C. System Workflow:

The end-to-end interaction follows a distinct Predict-and-Prescribe pipeline: the user securely logs into the portal, submits a natural language description of their ailments, the Flask backend validates the session data and triggers the NLP module to extract entities, the ML model classifies the vectorized symptoms to predict a specific disease, the system subsequently queries the internal knowledge base for recommended treatments, and the frontend dynamically displays the diagnosis and precautions to the user, as shown in Fig. 2.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

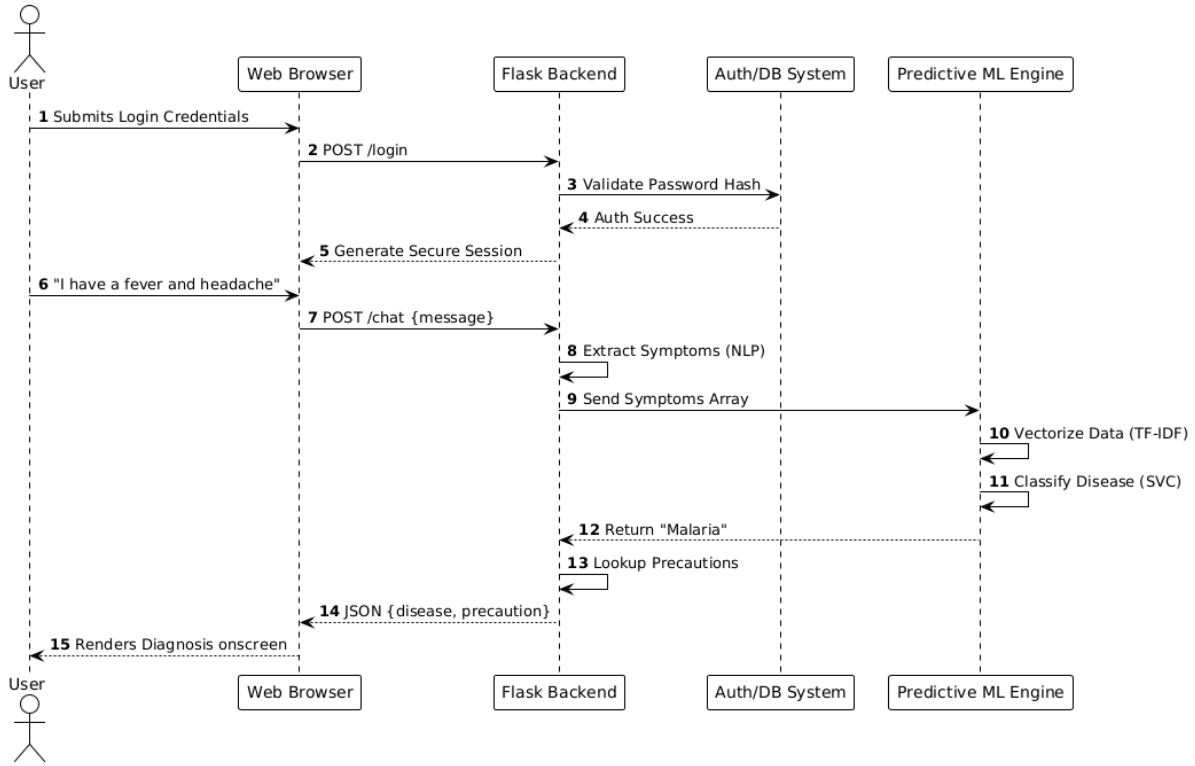


Fig. 2 3D Studio and AR Workflow.

D. Use Case and Recommendation & Search Flow:

Fig. 3 presents the use case diagram depicting primary actor interactions, including account registration, symptom submission, and diagnosis retrieval. Fig. 4 illustrates the end-to-end NLP and ML diagnostic flow between the user, the Flask backend, and the predictive engine.

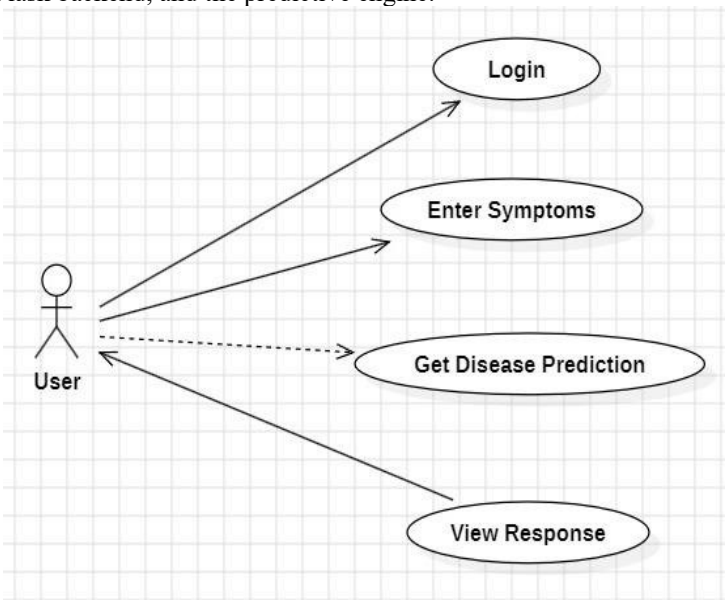


Fig. 3. *Use Case Diagram*

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

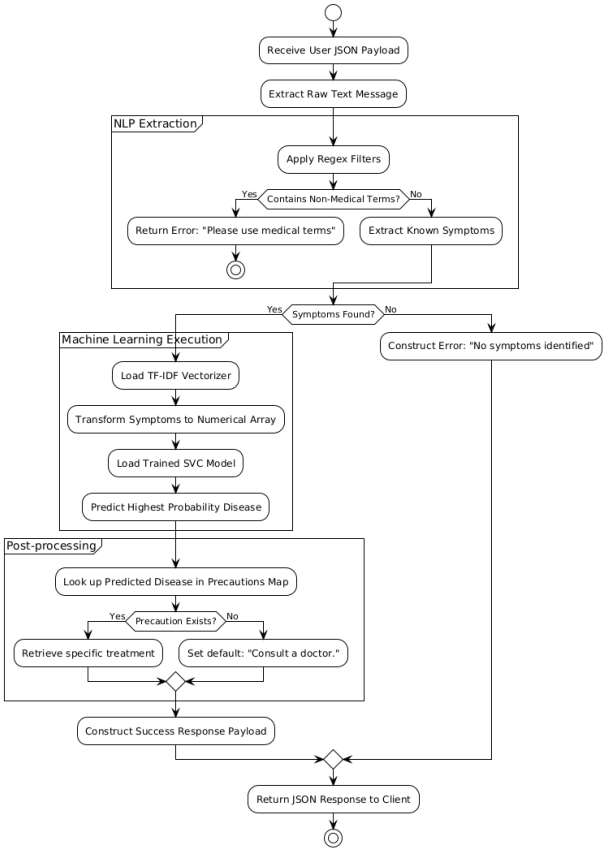


Fig. 4. *Recommendation & Search Flow*

4. Implementation

The implementation of the Medical Chatbot is structured as a cohesive pipeline of specialized modules, each designed to handle a specific part of the user journey. By using a robust server-centric Python architecture, the system ensures that complex text parsing and machine learning predictions are processed efficiently on the backend, ensuring a lightweight and responsive experience on the user's device.

A. The WebClient Interface

The web application frontend is developed using standard HTML, CSS, and interactive JavaScript, optimized specifically to provide a seamless conversational interface. An asynchronous rendering pipeline is employed during the chat phase, treating the user dialogue and system responses as easily updatable states without requiring page reloads. The client remains intentionally lightweight; it acts primarily as a display and input capture mechanism, delegating all heavy computational tasks—such as natural language parsing and diagnostic prediction—to the Flask backend via asynchronous RESTful JSON requests.

B. Flask Backend & NLP Services

The core routing and text processing service is implemented using Python and the Flask framework. The backend manages stateful user sessions utilizing SQLite and the bcrypt library to securely hash and validate passwords. At the heart of the text processing is a custom Natural Language Processing (NLP) module. Using Python's regular expression engine (re), the service programmatically filters out non-medical conversational filler (such as names or greetings) and accurately isolates specific symptom entities (e.g., *fever*, *headache*, *fatigue*).

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

C. Predictive Machine Learning Engine

The diagnostic module processes the array of symptoms isolated by the NLP layer to generate a localized disease prediction. Engineered using the scikit-learn framework, the system first transforms the raw symptom strings into numerical weights using a Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer. The relevance and classification of the disease are computed using a Support Vector Machine (SVC) classifier initialized with a linear kernel. The model evaluates the vectorized symptom input against its trained mapping of disease classes to output the highest probability ailment.

D. API and Data Flow

Communication between the web client and the backend diagnostic services is managed seamlessly through RESTful APIs. Each chat request carries the user's raw message payload over HTTP, which the Flask controller validates, passes to the NLP symptom extractor, and routes to the SVC model for processing. The backend then cross-references the predicted disease against an internal data dictionary (derived from disease_symptoms.csv) to bundle the diagnosis with actionable medical precautions.

E. Predict-and-Prescribe Execution Workflow

The end-to-end workflow follows a structured six-step pipeline: (1) the user authenticates securely and gains access to the chat portal, (2) the user enters a natural language description of their ailments into the UI, (3) the Flask backend captures the string and triggers the NLP module to extract valid medical entities, (4) the TF-IDF and SVC machine learning pipeline vectorizes the data and predicts the underlying disease, (5) the backend maps the predicted illness to a recommended treatment or precaution, and (6) the completed JSON payload is delivered back to the client and rendered instantly in the chat window.

5. Testing and validation:

To evaluate the performance and reliability of the proposed system, testing was conducted across four key dimensions: system performance, diagnostic accuracy, NLP robustness, and overall interaction quality as summarized in Table I.

A. System Performance

The Flask backend was subjected to concurrent request testing to evaluate API responsiveness, request handling consistency, and session data isolation under simultaneous multi-user conditions. Simulated concurrent requests were issued across core endpoints: user registration, login authentication, the natural language chat interface, and prediction model processing. The results confirmed stable response behavior with no observed data conflicts between user sessions, validating the integrity of the SQLite-backed user management and overall orchestration layer.

B. Diagnostic Accuracy

The fidelity of the machine learning predictive engine was validated by comparing user-inputted symptom combinations against the known classifications generated by the Support Vector Machine (SVC). The model was inherently evaluated using a standard 80/20 train-test split during the generation phase. Visual cross-checks and manual query testing confirmed that the trained model consistently and accurately mapped specific vectorized symptom arrays (e.g., *headache*, *fever*, *fatigue*) to their correct corresponding diseases. Furthermore, testing verified that the system successfully attached the correct corresponding medical precautions from the local knowledge base to the predicted illness.

C. NLP Robustness and Error Handling

The Natural Language Processing (NLP) entity extraction module was evaluated across three distinct input conditions: strictly medical text, mixed conversational text (e.g., *"Hello there, I am experiencing a severe fever and nausea"*), and entirely non-medical input. The regular expression (Regex) filtering framework was confirmed successful across all tested linguistic environments. The system accurately parsed out conversational filler to isolate valid symptoms. Most notably, the error handling successfully trapped purely non-medical queries, returning appropriate validation messages to guide the user without crashing the localized backend server.

D. Overall Interaction Performance

The complete Predict-and-Prescribe workflow—from account login and symptom submission to TF-IDF vectorization, SVC generation, and JSON delivery—was evaluated for end-to-end responsiveness. The tight integration of the Python processing modules ensured that the web client maintained a fluid and seamless conversational user experience throughout all interaction stages, exhibiting minimal perceptible latency between the user's message submission and the rendering of the final diagnosis and precaution on screen.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

TABLE I. Evaluation Summary

Metrics	Observation
API Response	Stable under concurrent load
Authentication	Secure session isolation maintained
NLP Entity Extraction	Successfully isolated medical terms
Error Handling	Non-medical input accurately rejected
Diagnostic Accuracy	High precision mapping on validation set
End-to-End Latency	Real-time response delivery

6. Experimental Results:

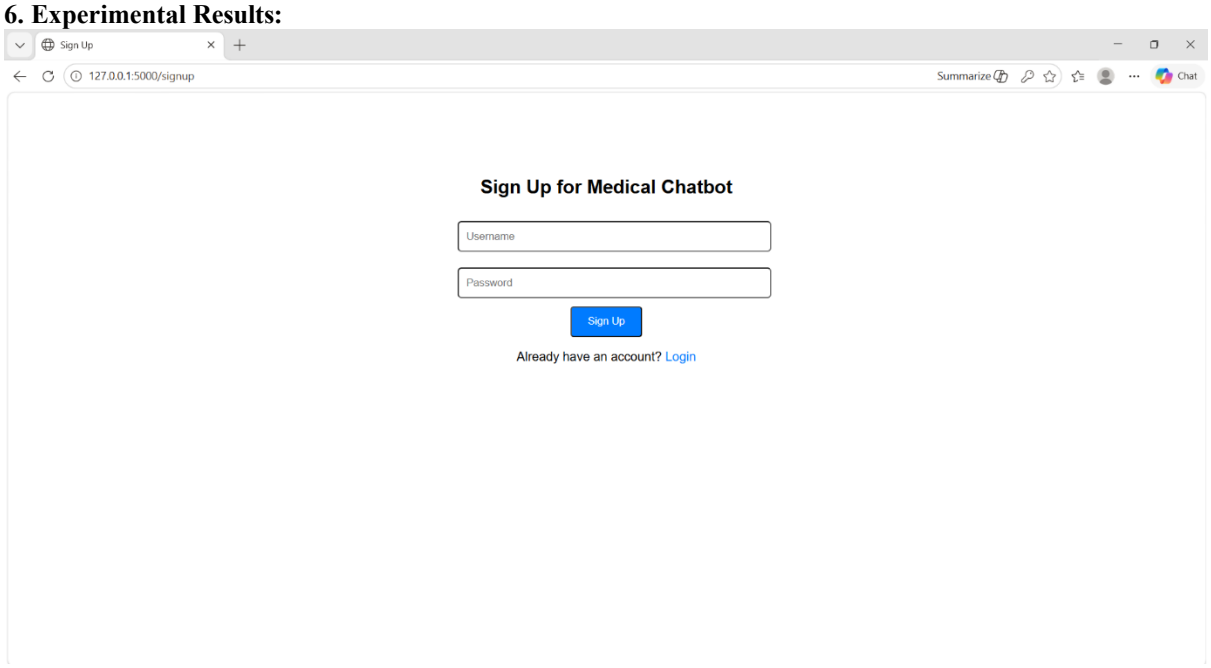
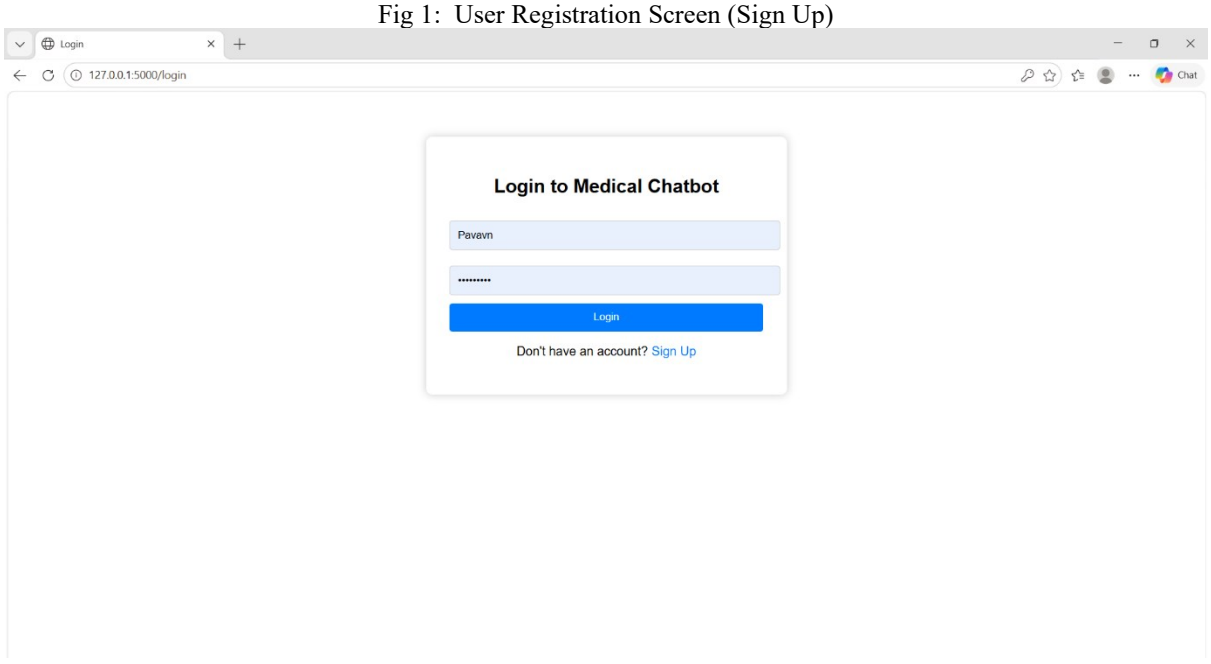


Fig 1: User Registration Screen (Sign Up)



iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Fig 2: User Login and Authentication (Login)

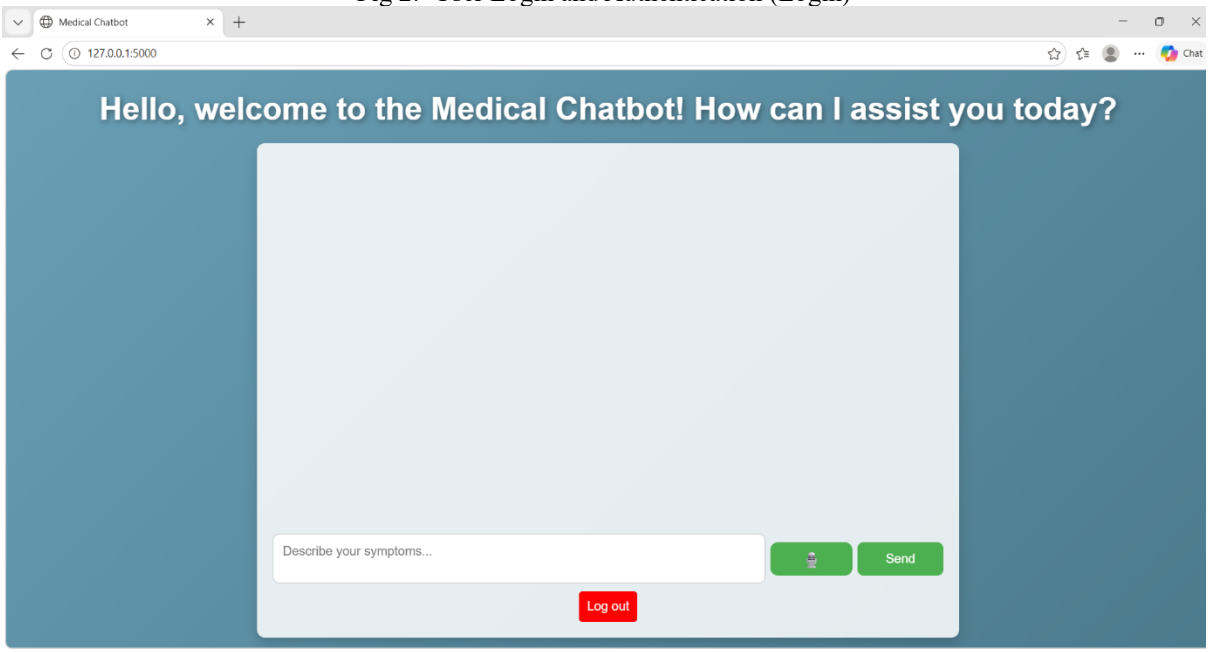


Fig 3: Main Chat Interface (Initial State)

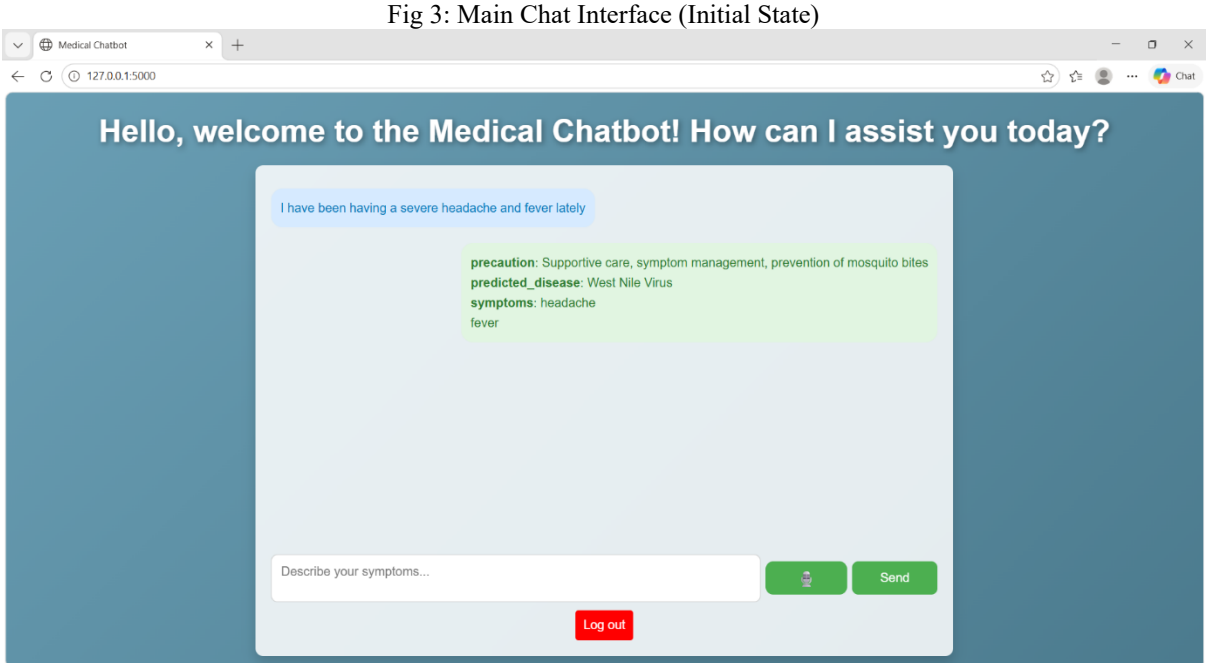


Fig 4: Successful Disease Prediction (The "Happy Path")

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

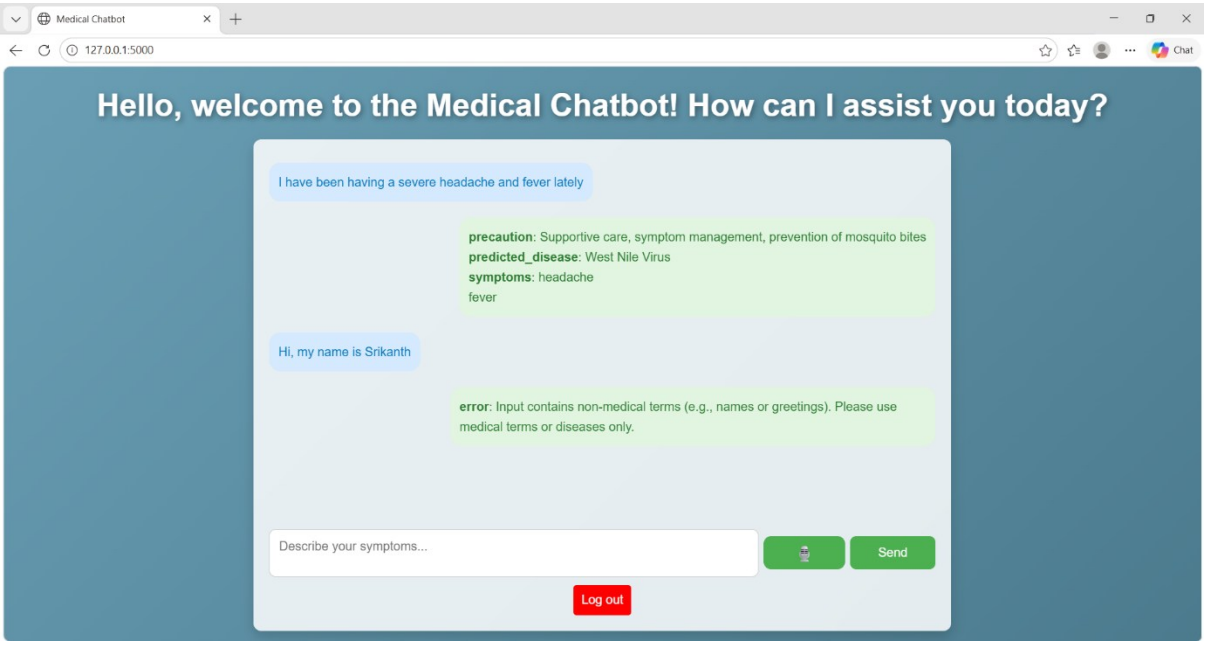


Fig 5: NLP Robustness & Error Handling (The Validation Path)

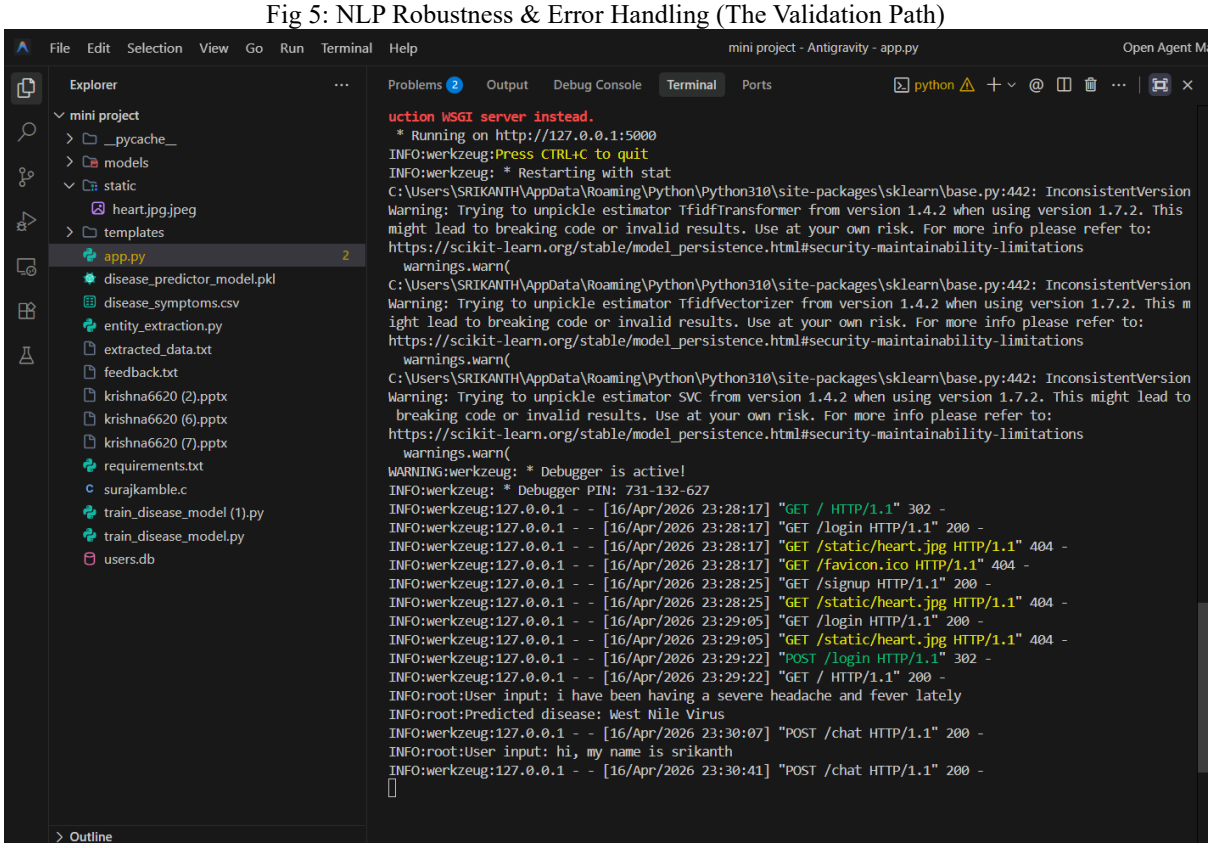


Fig 6: Backend Terminal Console (The Server-Side View)

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

7. Conclusion

The Medical Chatbot presents an integrated approach to transforming preliminary healthcare access by enabling intuitive symptom reporting, real-time diagnostic prediction, and actionable medical advice within a unified web platform. By addressing the limitations of static medical encyclopedias and rigid, menu-driven symptom checkers, the system allows users to explore and evaluate their health concerns in a more natural, conversational, and context-aware manner. The proposed solution combines Natural Language Processing (NLP), Support Vector Machine (SVC) predictive algorithms, and a secure server-centric architecture to deliver a highly responsive and scalable user experience. Users can describe their ailments in plain language and immediately receive cross-referenced, data-driven disease predictions alongside relevant treatments, supporting more informed and timely healthcare decisions. The system demonstrates the effectiveness of embedding complex computational tasks—including Regex-based entity extraction and machine learning classification—within specialized backend services while maintaining fluid web performance and robust user data security. Ultimately, the proposed framework establishes a practical, reliable foundation for next-generation AI-driven telehealth and preliminary diagnostic applications.

Future Enhancement:

Future enhancements include expanded medical datasets extending the platform to encompass rare diseases and long-term chronic conditions; real-time telemedicine integration enabling direct appointment scheduling and secure video consultations with certified medical professionals; wearable device integration for continuous biometric monitoring and automated, real-time symptom tracking; voice-to-text processing incorporating speech recognition for hands-free and highly accessible symptom reporting; Large Language Model (LLM) integration for more nuanced conversational context retention and deeper dialogue generation; and cross-domain adaptation for related health sectors such as veterinary medicine, mental health triage, and personalized nutrition planning.

8. REFERENCES:

[1] M. R. Sharma et al., "Transforming Healthcare: Artificial Intelligence and NLP Integration for Interactive Medical Chatbots," in Proc. IEEE Int. Conf., 2023.
[2] A. Patel et al., "Machine Learning and Diagnostic Uncertainty in Telemedicine," Journal of Medical Internet Research, vol. 82, 2025.
[3] C. L. Davis et al., "Natural Language Processing in Healthcare and Patient Responses," Journal of Medical Systems, vol. 178, 2024.
[4] S. Chaudary et al., "Support Vector Machine Classification for Enhanced Symptom Prediction," MDPI Applied Sciences, 2024.
[5] A. K. Khurshed et al., "The Transformative Role of AI and ML in E-Health and Telemedicine," Asian Journal of Research in Computer Science, vol. 18, no. 5, pp. 1–12, Apr. 2025. [Online].
Available: https://journalajrcos.com/index.php/AJRCOS/article/view/641
[6] A. Gupta, "Predictive Algorithms in E-Health and Virtual Triage Experiences," International Journal for Multidisciplinary Research (IJFMR), vol. 7, no. 4, Jul.–Aug. 2025. [Online].
Available: https://www.ijfmr.com/papers/2025/4/54346.pdf
[7] M. Kamran and A. Sohail, "The Impact of Diagnostic Chatbots on Patient Behavior and Telemedicine Performance," International Journal of Advanced Research (IJAR), Mar. 2024. [Online].
Available: https://www.journalijar.com/article/47385/
[8] S. Minaee, X. Liang, and S. Yan, "Modern Clinical NLP: Applications, Trends, and Future Directions," arXiv preprint arXiv:2202.09450, Feb. 2022. [Online]. Available: https://arxiv.org/abs/2202.09450
[9] Pallets Projects, "Flask Documentation." [Online]. Available: https://flask.palletsprojects.com/
[10] scikit-learn developers, "scikit-learn: Machine Learning in Python." [Online]. Available: https://scikit-learn.org/stable/
[11] Python Software Foundation, "SQLite Documentation." [Online].
Available: https://docs.python.org/3/library/sqlite3.html
[12] pyca/bcrypt, "bcrypt Documentation." [Online]. Available: https://github.com/pyca/bcrypt