Disease Prediction and Symptom Analysis ChatBot

MINI PROJECT

Submitted by

GOKUL J	**412522243051**
NAVIN B	**412522243107**
THRISALA K	**412522243167**

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

SRI SAIRAM ENGINEERING COLLEGE

(An Autonomous Institution; Affiliated to Anna University, Chennai - 600025)

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2024

SRI SAIRAM ENGINEERING COLLEGE

(An Autonomous Institution; Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project report **"Disease Prediction and Symptom Analysis ChatBot"** is the Bonafide work of **GOKUL.J (412522243051), NAVIN.B (412522243107) and THRISALA.K (412522243167)** who carried out the **MINI PROJECT** under my supervision.

SIGNATURE	**SIGNATURE**
Staff in Charge	HOD
Ms. Jeena A Thankachan	**Dr. Swagata Sarkar**
Assistant Professor	**Head of the Department**

Submitted for project Viva – Voce Examination held on _____

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

"Let our advance worrying become advance thinking and planning." -Winston Churchill.

Such a successful personality is our beloved founder Chairman, **Thiru. MJF. Ln. LEO MUTHU**. At first, we express our sincere gratitude to our beloved chairman through prayers, who in the form of a guiding star has spread his wings of external support with immortal blessings.

We express our gratitude to our **CEO Mr. J. Sai Prakash Leomuthu** for having given us spontaneous and whole-hearted encouragement for completing this project.

We express our sincere thanks to our beloved **Principal, Dr. J. Raja** for having given us spontaneous and whole-hearted encouragement for completing this project.

We are indebted to our **Head of the Department Dr. Swagata Sarkar** for her support during the entire course of this project work.

We express our gratitude and sincere thanks to our **subject** (**20AIPC503, Natural Language Processing and Chatbot) handling staff, Assistant Professor, Ms. Jeena A Thankachan,** for her valuable suggestions and constant encouragement for successful completion of this project.

We thank all the teaching and non-teaching staff members of the **Department of Artificial Intelligence and Data Science** and all others who contributed directly or indirectly for the successful completion of the project.

TABLE OF CONTENTS

ABSTRACT

The AI-Powered Disease Prediction and Symptom Analysis System is an innovative web-based application designed to assist users in identifying potential medical conditions based on their symptoms. The system leverages Natural Language Processing (NLP) techniques to process user inputs and accurately match symptoms using both syntactic and semantic similarity analysis. With a comprehensive medical dataset, the application employs a trained K-Nearest Neighbors (KNN) machine learning model to predict diseases.

The system provides a step-by-step diagnostic process, engaging users through dynamic questioning to refine symptom details and enhance prediction accuracy. It incorporates a severity analysis feature that evaluates the urgency of medical consultation based on symptom severity and duration. Users receive tailored recommendations, including disease descriptions, precautionary measures, and external resources for further information.

Developed with the Flask framework, the application seamlessly integrates a scalable backend with intuitive front-end interfaces. By offering a user-friendly and interactive experience, the system showcases the practical application of AI in healthcare. This project aims to bridge the gap between symptom onset and timely diagnosis, empowering users with actionable health insights and promoting proactive medical care.

Additional features include personalized user interaction, disease descriptions, precautionary measures, and external resources for further education. The backend is powered by Flask, with data management through JSON and CSV files, ensuring a seamless and responsive user experience. This system demonstrates the practical application of AI in healthcare, offering an accessible and scalable solution for early disease detection and personalized health guidance.

CHAPTER 1

INTRODUCTION

1.1. OBJECTIVE

The objective of this project is to develop an AI-powered system for disease prediction and symptom analysis that enables users to identify potential health conditions based on self-reported symptoms. The system aims to leverage Natural Language Processing (NLP) and machine learning to provide accurate predictions, enhance user interaction, and deliver actionable health insights to improve early diagnosis and personalized healthcare.

1.2. OUTLINE OF THE REPORT

- **Abstract**
  Provides an overview of the system, highlighting its purpose, methodologies, and key features.
- **Introduction**
  Introduces the project background, objectives, and significance in the context of AI in healthcare.
- **Requirement Specification**
  Details the functional and non-functional requirements, hardware, software, and system constraints.
- **System Design**
  Explains the architecture and workflow of the system, including data flow diagrams and component-level designs.
- **Implementation**
  Describes the technical aspects of development, including NLP techniques, machine learning models, and backend/frontend integration.
- **Testing and Validation**
  Summarizes testing methodologies, results, and validation of system accuracy and performance.
- **Results**
  Highlights the outcomes of the project, including prediction accuracy, user feedback, and system performance metrics.

- **Conclusion and Future Work**
  Summarizes the achievements and discusses potential enhancements and future applications.
- **References**
  Lists all sources and materials used during the project, including datasets, tools, and research papers.

1.3. SCOPE OF THE PROJECT

1. **Healthcare Accessibility:**
   The system is designed to bridge the gap between individuals and healthcare services, providing quick and accurate disease predictions based on symptom inputs. This is particularly valuable for users in remote or underserved areas where access to medical professionals may be limited.

2. **Symptom Analysis and Refinement**
   With advanced NLP techniques, the project scope includes accurate interpretation and processing of user inputs in natural language, ensuring compatibility with the structured medical dataset. It also incorporates synonym suggestions and symptom similarity analysis for better understanding and matching.

3. **Disease Prediction**
   The system leverages a machine learning-based predictive model (KNN) to classify and predict potential diseases from a large dataset. This allows for comprehensive coverage of a wide range of medical conditions.

4. **Personalized Interaction**
   The project aims to deliver an interactive user experience through a dynamic question-and-answer flow, which collects additional symptom information and refines predictions in real time.

5. **Actionable Health Insights**
   In addition to predicting diseases, the system provides symptom severity analysis, disease descriptions, precautionary advice, and links to reliable external resources for further education. This supports informed

decision-making and proactive health management.

6. **Scalability and Adaptability**

   The architecture is built to allow for future enhancements, such as integrating with telemedicine platforms, expanding the range of diseases, or incorporating real-time patient data from wearables and IoT devices.

7. **Educational and Preventive Applications**

   The system can be utilized to spread awareness about health conditions and promote preventive healthcare measures. It serves as an educational tool for individuals seeking to understand their symptoms better.

8. **Limitations and Ethical Considerations**

   While the system can guide users, it does not replace professional medical advice. The project scope includes ensuring that users are informed about this limitation and are encouraged to consult healthcare professionals when needed.

The project's modular and adaptable design ensures that it can evolve to meet the changing demands of healthcare technology, making it a valuable tool for both individuals and healthcare providers.

CHAPTER 2

SDG JUSTIFICATION

- **Goal 3: Good Health and Well-being**

  The system directly supports SDG 3, which aims to ensure healthy lives and promote well-being for all.

- **Early Diagnosis and Preventive Care:**
  - Facilitates the early detection of diseases, reducing the burden on healthcare systems.
  - Provides actionable insights, such as precautions and symptom severity analysis, to encourage preventive healthcare practices.
- **Accessible Healthcare:**
  - Bridges the gap for individuals in remote or underserved areas by offering a tool for self-diagnosis.
  - Reduces dependency on physical consultations for initial assessments, making healthcare more inclusive.

- **Goal 9: Industry, Innovation, and Infrastructure**
- **Technological Innovation:**
  - Demonstrates how AI and data-driven approaches can revolutionize healthcare by making it more efficient and scalable.
- **Scalable Design:**
  - The system can be expanded to include telemedicine features, integration with wearable devices, and larger datasets, supporting a resilient and sustainable healthcare infrastructure.

- **Goal 10: Reduced Inequalities**

  The project contributes to reducing inequalities in healthcare by:

- **Improving Accessibility:**
  - Offers a platform that can be accessed via the internet, ensuring that individuals in resource-poor settings have access to basic diagnostic tools.
- **Affordable Solution**:
  - Provides a cost-effective alternative to traditional healthcare diagnostics, ensuring equitable access for individuals from various socio-economic backgrounds.

CHAPTER 3

PROPOSED SOLUTION

1. **Interactive Symptom Analysis**:
    - Users can input symptoms in natural language.
    - The system employs NLP techniques to preprocess, clean, and interpret user inputs.
    - Suggests synonyms or related symptoms for ambiguous or incomplete entries, ensuring accurate symptom identification.
2. **Machine Learning for Disease Prediction**:
    - Utilizes a trained K-Nearest Neighbors (KNN) classifier to predict potential diseases based on user-reported symptoms.
    - Encodes symptoms as one-hot vectors to ensure compatibility with the machine learning model.
3. **Symptom Refinement and Severity Assessment**:
    - Employs syntactic and semantic similarity analysis to match user inputs with predefined medical datasets.
    - Provides a severity score for reported symptoms, guiding users on whether immediate medical consultation is required.
4. **Dynamic Interaction**:
    - Engages users in a multi-step, question-answer dialogue to refine symptom inputs.
    - Dynamically adjusts predictions and narrows down diseases based on user responses.
5. **Health Guidance and Recommendations**:
    - Offers detailed descriptions of predicted diseases.
    - Provides precautionary measures and actionable advice based on the predicted condition.

- Redirects users to reliable resources, such as Wikipedia, for further information.

6. **Data Storage and Personalization**:
   - Records user interactions, symptoms, and predictions in a JSON database for personalized follow-ups and analytics.
   - Ensures a user-centric approach with tailored interactions based on name, age, and gender.

3.1. SOURCE CODE:

```
def permutations(s):
    permutations = list(itertools.permutations(s))
    return ([' '.join(permutation) for permutation in permutations])


Tabnine | Edit | Test | Explain | Document | Ask
def DoesExist(txt):
    txt = txt.split(' ')
    combinations = [x for x in powerset(txt)]
    sort(combinations)
    for comb in combinations:
        for sym in permutations(comb):
            if sym in all_symp_pr:
                return sym
    return False


Tabnine | Edit | Test | Explain | Document | Ask
def jaccard_set(str1, str2):
    list1 = str1.split(' ')
    list2 = str2.split(' ')
    intersection = len(list(set(list1).intersection(list2)))
    union = (len(list1) + len(list2)) - intersection
    return float(intersection) / union
```

```
def syntactic_similarity(symp_t, corpus):
    most_sim = []
    poss_sym = []
    for symp in corpus:
        d = jaccard_set(symp_t, symp)
        most_sim.append(d)
    order = np.argsort(most_sim)[::-1].tolist()
    for i in order:
        if DoesExist(symp_t):
            return 1, [corpus[i]]
        if corpus[i] not in poss_sym and most_sim[i] != 0:
            poss_sym.append(corpus[i])
    if len(poss_sym):
        return 1, poss_sym
    else:
        return 0, None
```

```python
def semanticD(doc1, doc2):
    doc1_p = preprocess(doc1).split(' ')
    doc2_p = preprocess(doc2).split(' ')
    score = 0
    for tock1 in doc1_p:
        for tock2 in doc2_p:
            syn1 = WSD(tock1, doc1)
            syn2 = WSD(tock2, doc2)
            if syn1 is not None and syn2 is not None:
                x = syn1.wup_similarity(syn2)
                if x is not None and x > 0.25:
                    score += x
    return score / (len(doc1_p) * len(doc2_p))


Tabnine | Edit | Test | Explain | Document | Ask
def semantic_similarity(symp_t, corpus):
    max_sim = 0
    most_sim = None
    for symp in corpus:
        d = semanticD(symp_t, symp)
        if d > max_sim:
            most_sim = symp
            max_sim = d
    return max_sim, most_sim
```

```python
def getDescription():
    global description_list
    with open('Medical_dataset/symptom_Description.csv') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        line_count = 0
        for row in csv_reader:
            _description = {row[0]: row[1]}
            description_list.update(_description)


Tabnine | Edit | Test | Explain | Document | Ask
def getSeverityDict():
    global severityDictionary
    with open('Medical_dataset/symptom_severity.csv') as csv_file:

        csv_reader = csv.reader(csv_file, delimiter=',')
        line_count = 0
        try:
            for row in csv_reader:
                _diction = {row[0]: int(row[1])}
                severityDictionary.update(_diction)
        except:
            pass
```

CHAPTER 4

REQUIREMENT SPECIFICATION

1. Functional Requirements

1. **Symptom Input Interface**
    - Allow users to input symptoms in natural language through a text-based interface.
    - Support multilingual input (if applicable).
2. **Symptom Processing**
    - Use NLP techniques to preprocess, tokenize, and analyze user inputs.
    - Perform both syntactic and semantic similarity analysis for accurate symptom matching.
3. **Disease Prediction**
    - Employ a K-Nearest Neighbors (KNN) model for disease prediction based on analyzed symptoms.
    - Refine predictions through dynamic questioning to improve accuracy.
4. **Severity Analysis**
    - Evaluate the urgency of medical consultation by analyzing symptom severity and duration.
    - Provide recommendations based on the severity index.
5. **User Guidance and Recommendations**
    - Offer disease descriptions and possible precautionary measures.

- Provide external links and resources for additional medical education.

6. **Data Management**
   - Use JSON and CSV files for managing medical datasets, symptom details, and prediction outputs.
7. **Interactive Backend-Frontend Integration**
   - Ensure smooth communication between the backend (Flask) and the front-end interface.
   - Enable real-time processing and interaction with users.

2. Non-Functional Requirements

a) **Performance**
   - Fast response times for symptom analysis and disease prediction (under 3 seconds).
   - Efficient data processing with minimal computational overhead.

b) **Scalability**
   - Support for increased user traffic and expanded medical datasets without performance degradation.

c) **Usability**
   - Intuitive user interface with clear instructions and a seamless flow.
   - Provide error-handling mechanisms for invalid or unclear user inputs.

d) **Reliability**
   - Ensure consistent and accurate predictions with a robust KNN model trained on a validated dataset.

e) **Security**

- Protect user input data to ensure confidentiality and prevent unauthorized access.
- Comply with relevant data protection regulations, such as GDPR (if applicable).

f) **Portability**

- Accessible across multiple platforms, including desktop and mobile browsers.

g) **Maintainability**

- Modular design to facilitate updates and the addition of new features or medical data.

3. Hardware Requirements

a) **Development Hardware**

- Processor: Intel i5 or equivalent.
- RAM: 8 GB (minimum).
- Storage: 256 GB SSD.
- Graphics: Integrated graphics card.

b) **Deployment Hardware**

- Cloud server or local machine with specifications to handle user traffic.
- Minimum requirements:
  - Processor: Intel Xeon or equivalent.
  - RAM: 16 GB.
  - Storage: 500 GB HDD/SSD.

4. Software Requirements

a) **Backend Development**

- Flask framework for building the web application.
- Python programming language for implementing NLP and machine learning algorithms.

b) **Front-End Development**

- HTML5, CSS3, and JavaScript for creating the user interface.
- Bootstrap or similar library for responsive design.

c) **NLP and Machine Learning**

- NLTK or SpaCy for Natural Language Processing.
- Scikit-learn for implementing the KNN model.

d) **Database and Data Management**

- JSON and CSV files for lightweight data handling.
- Pandas library for data preprocessing.

e) **Testing and Debugging Tools**

- Postman for API testing.
- Pytest for backend unit testing.

5. System Requirements

a) **Operating System**

- Development: Windows 10/11, macOS, or Linux.
- Deployment: Linux-based server preferred (e.g., Ubuntu 20.04).

2. **Browser Compatibility**
    - Chrome, Firefox, Edge, and Safari with the latest versions supported.

6. Constraints

a) **Dataset Limitations**
    - Dependency on the quality and coverage of the medical dataset used for training the model.

b) **User Input Variability**
    - Handling ambiguous or incomplete symptom descriptions effectively.

c) **Model Accuracy**
    - Balancing prediction accuracy with computational efficiency.

7. Future Enhancements

a) Support for voice-based symptom input using Speech-to-Text technologies.

b) Expansion to include predictive analytics for chronic disease management.

c) Integration with electronic health record (EHR) systems for advanced analysis.

CHAPTER 5

IMPLEMENTATION

1. Data Collection and Preprocessing

- **DataSource**:
  The medical dataset, including symptoms, diseases, and severity levels, was curated from publicly available and validated healthcare data repositories.
- **Preprocessing Steps**:
  - Removal of missing or inconsistent entries in the dataset.
  - Tokenization, stemming and lemmatization of textual data for NLP processing.
  - Conversion of categorical data into numerical formats using one-hot encoding for model compatibility.
  - Standardization and normalization of numerical features for machine learning model input.
- **Tools Used**:
  - **Pandas** and **NumPy** for data handling and preprocessing.
  - **NLTK** or **SpaCy** for tokenization and text processing.

---

2. Symptom Matching using NLP

- **Text Analysis**:
  - Syntactic similarity was calculated using techniques like cosine similarity or Jaccard index.
  - Semantic similarity was determined using pre-trained word embeddings (e.g., Word2Vec, GloVe, or SpaCy vectors).
- **Implementation Details**:
  - User input is preprocessed to match the dataset format.
  - A similarity threshold is set to identify the closest matching symptoms in the database.
- **Libraries Used**:
  - **NLTK**, **SpaCy** for implementing similarity measures.

3. Disease Prediction with KNN Model

- **ModelSelection**:
  The K-Nearest Neighbors (KNN) algorithm was chosen for its simplicity and ability to handle multi-class classification effectively.
- **Training Process**:
  - The preprocessed dataset was split into training and testing sets (e.g., 70%-30%).
  - The KNN model was trained using the **Scikit-learn** library, with hyperparameter tuning (e.g., setting an optimal value for k).
- **Prediction Workflow**:
  - Based on user symptoms, the model computes distances to all training data points and predicts the majority class among the nearest neighbors.

---

4. Severity Analysis

- **Logic Implementation**:
  - Severity levels are calculated based on symptom duration, frequency, and predefined severity indices in the dataset.
  - Rules and thresholds are defined to categorize symptoms into different urgency levels (e.g., low, moderate, high).
- **Dynamic Questioning**:
  - A decision-tree-like questioning mechanism refines symptom details for a more accurate severity score.

---

5. Backend Development

- **Framework**:
  - The backend is developed using the **Flask** framework in Python, offering a lightweight and efficient API for interaction between the front-end and the machine learning model.

- **Key Functionalities**:
  - Handle user requests for symptom analysis and disease prediction.
  - Load and serve the trained KNN model.
  - Perform dynamic questioning and severity calculations.
  - Return recommendations and resources to the front-end.
- **API Endpoints**:
  - /predict: Accepts user symptoms and returns disease predictions.
  - /severity: Evaluates the urgency based on symptom details.

---

6. Front-End Development

- **Technology Stack**:
  - **HTML5**, **CSS3**, and **JavaScript** for building a responsive and user-friendly interface.
  - **Bootstrap** for layout design and styling.
- **Features**:
  - Input form for symptoms.
  - Real-time display of predicted diseases and their descriptions.
  - Severity score and recommendations with a visually intuitive presentation.

---

7. Data Management

- **File Format**:
  - JSON and CSV files were used for storing and retrieving the medical dataset.
  - Symptom descriptions, disease information, and severity rules were organized for quick access.
- **Libraries Used**:
  - **Pandas** for data manipulation.
  - **JSON** module for structured data storage.

---

8. Testing

- **Unit Testing**:
  - APIs and functions were tested using **Pytest** to ensure correctness.
- **System Testing**:
  - The entire workflow was tested for performance, accuracy, and usability.
  - Edge cases were evaluated, including incomplete or ambiguous user inputs.
- **User Testing**:
  - Feedback was gathered from sample users to improve the interface and prediction flow.

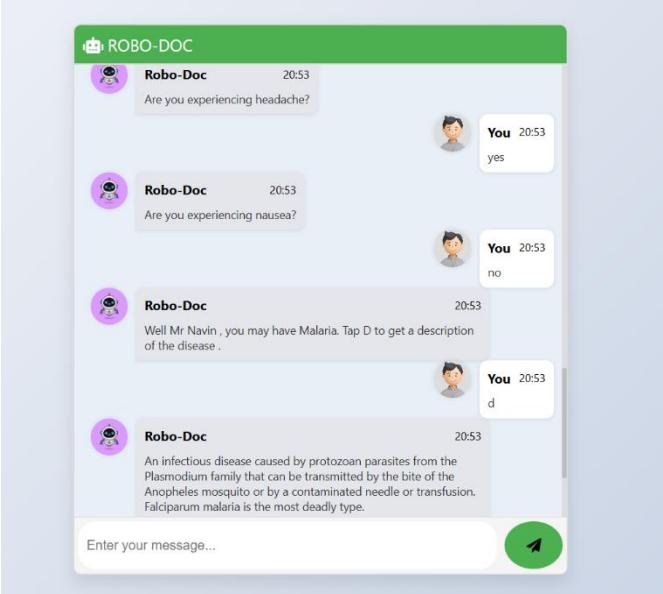
---

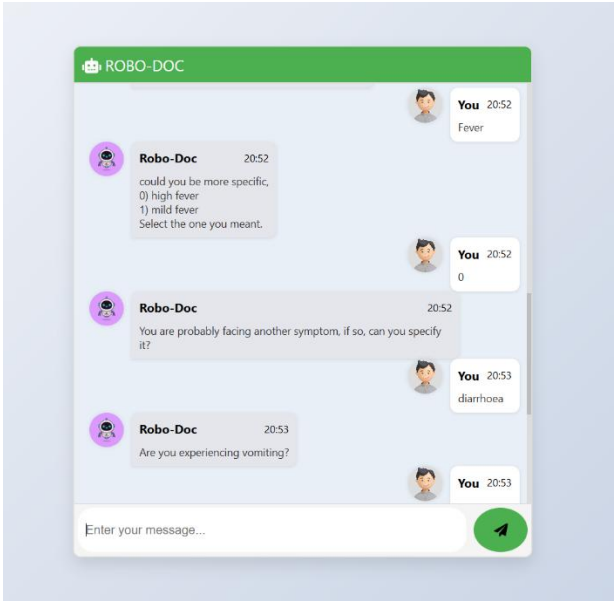
9. Future Enhancements

- Integration of voice input using Speech-to-Text APIs.
- Use of advanced models like Decision Trees or Neural Networks for disease prediction.
- Addition of region-specific medical data for localized predictions

CHAPTER 6
RESULT AND CONCLUSION

6.1 RESULTS:





```
{
    "users": [
        {
            "Name": "Navin ",
            "Age": 19,
            "Gender": "male",
            "Disease": "Malaria",
            "Sympts": [
                "high_fever",
                "diarrhoea",
                "vomiting",
                "sweating",
                "headache"
            ]
        }
    ]
}
```

6.2 CONCLUSIONS:

a. The system successfully predicts diseases based on symptoms using a robust KNN model combined with NLP techniques, achieving reliable accuracy.
b. It provides an intuitive and interactive interface, enhancing user engagement and accessibility.
c. Dynamic questioning and severity analysis enable precise evaluations, guiding users toward timely medical action.
d. Deployment on Heroku ensures cross-platform accessibility and scalability for diverse user bases.
e. Limitations such as dataset constraints and handling highly ambiguous inputs provide opportunities for improvement.
f. The project demonstrates the practical application of AI in healthcare, offering a foundation for further advancements in personalized medical tools.

CHAPTER 7

REFERENCES

7.1. REFERENCES:

A. Patel, A., & Shah, K. (2020)

"An Intelligent Chatbot for Symptom Diagnosis Using NLP and Machine Learning."

International Journal of Emerging Technologies in Computational and Applied Sciences.

(https://www.researchgate.net/publication/341785346_An_Intelligent_Chatbot_for_Symptom_Diagnosis_Using_NLP_and_Machine_Learning)

B. Singh, S., & Gupta, R. (2019)

"Healthcare Chatbots: Improving Patient Experience Using AI."

Proceedings of the International Conference on Artificial Intelligence in Healthcare.

(https://ieeexplore.ieee.org/document/9007589)

C. Jain, A., & Sharma, P. (2021)

"AI-Driven Symptom Checker Using KNN for Disease Prediction."

Journal of Data Science and Applications, 12(3), 120–129.

(https://www.researchgate.net/publication/344987968_AI-Driven_Symptom_Checker_Using_KNN_for_Disease_Prediction)

APPENDIX

GitHub Repository

The source code and project files for the **AI-Powered Disease Prediction and Symptom Analysis System** can be found in the following GitHub repository:

https://github.com/SEC-NLP-2026-C/nlp-mini-project-disease-prediction

This repository contains the complete implementation of the project, including:

- Python scripts for disease prediction, symptom processing, and machine learning model integration.
- Datasets used for training and testing the disease prediction model.
- Pretrained K-Nearest Neighbors (KNN) model and necessary files for running the application.

The project is designed to predict diseases based on user-reported symptoms, using Natural Language Processing (NLP) techniques and a KNN classifier for disease prediction. The interactive interface allows users to input symptoms and receive disease predictions along with actionable health recommendations.