Disease Detection System

GROUP NO.-174

Project Exhibition -1 (DSN2098)

Supervisor:-
Dr. Ajeet Singh

MEMBERS

Shivam Krishna – 23BCE10589
Kanishk Tomar – 23BCE10588
Ankit Kumar – 23BCE10519
Myisha Porwal – 23BCE11519
Stuti S. Agrawal – 23BCE11717

Reviewer:
Dr.Mohammad Sultan Alam
Dr.Shafiul Amol Ahmed

INTRODUCTION

The Disease Detecting System is designed to predict potential diseases based on user-provided symptoms. This project aims to assist users in identifying underlying health issues by leveraging advanced technologies like machine learning. With features like a user-friendly interface, voice and text-based symptom input, and visually appealing results, this project sets a new benchmark in healthcare accessibility and efficiency.

CHALLENGES IN EXISTING SYSTEMS

- Limited Accuracy:
  - Many existing systems rely on simple rule-based algorithms or symptom-to-disease mapping, which often fail to provide accurate predictions in complex cases or when symptoms overlap between multiple diseases.
- Lack of Personalization:
  - These systems usually do not account for individual variations, such as age, gender, medical history, or lifestyle factors, which can significantly impact disease prediction.
- No Severity Analysis:
  - Existing systems often do not assess or communicate the severity of the predicted diseases, leaving users with limited actionable insights.
- Language Barriers:
  - Most systems are available only in a single language, making them inaccessible to a diverse population.

PROPOSED WORK AND METHODOLOGY

Transition to Machine Learning (ML):

01.

Train models to predict diseases based on symptom patterns.

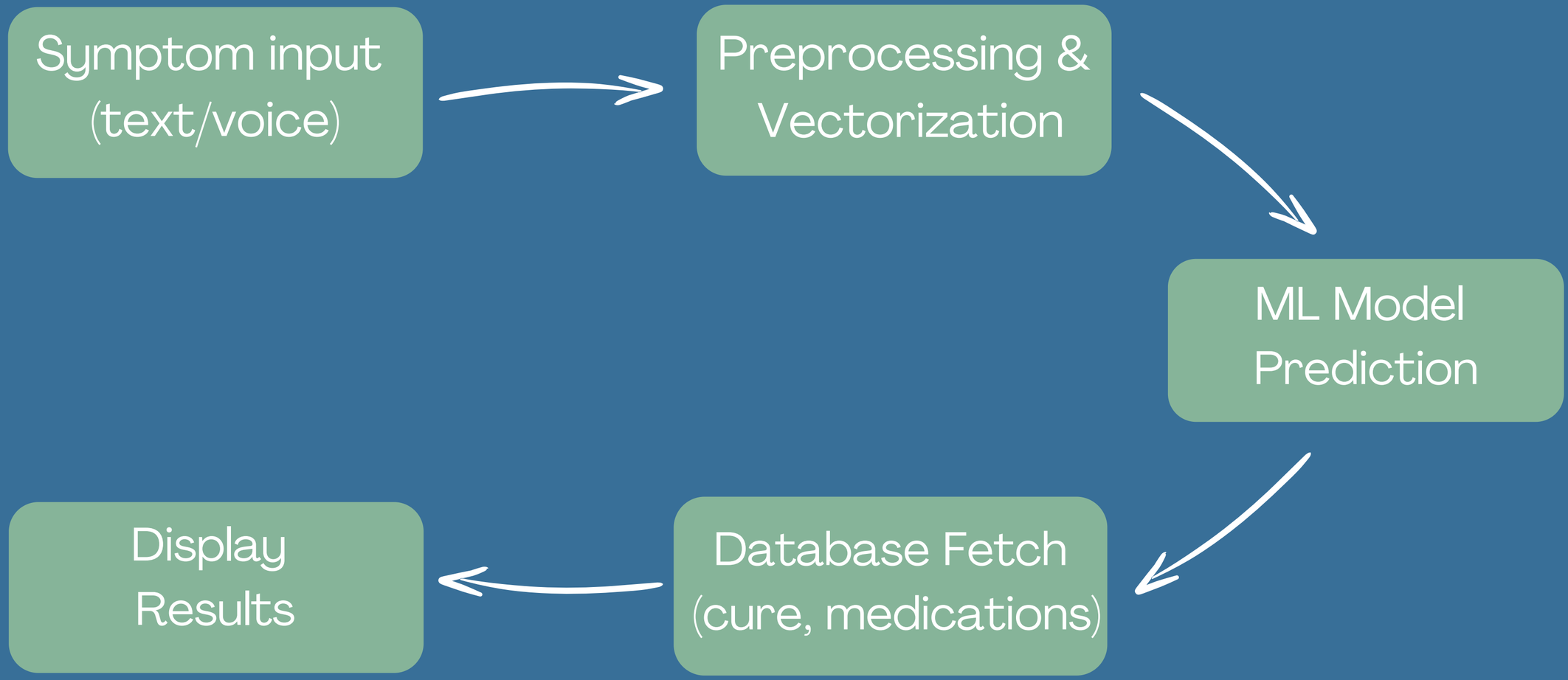
02.

Use Scikit-learn tools for training and CountVectorizer for text preprocessing.

03.

Incorporate a database for detailed disease information.

Methodology Flowchart

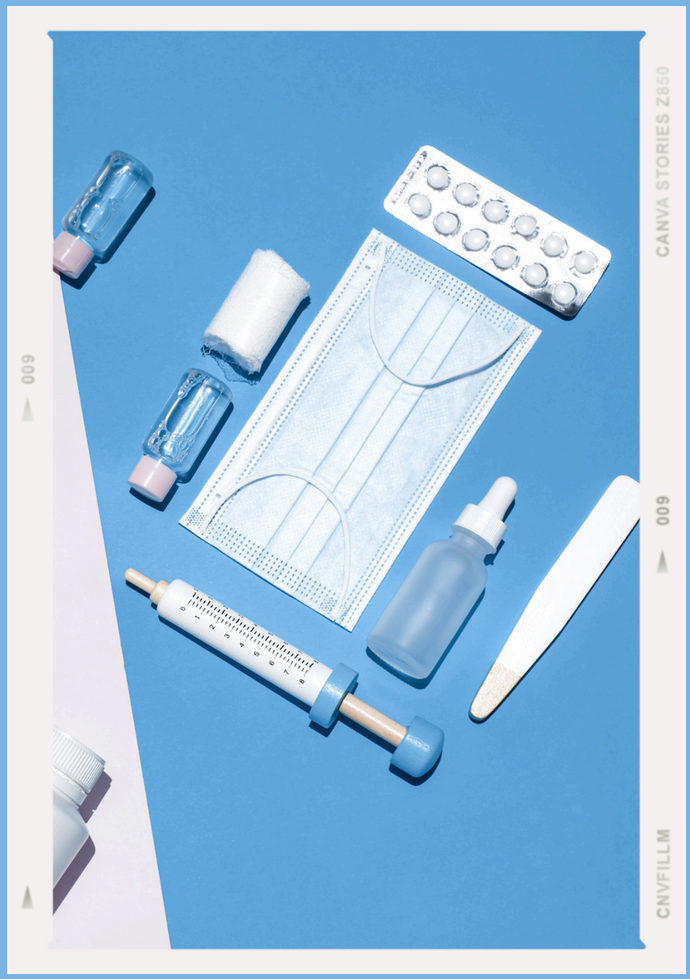


KEY FEATURES OF THE PROJECT

- User-Friendly Input
  - Allows users to input symptoms through text or voice for ease of use.
- Symptom Preprocessing and Vectorization
  - Utilizes data preprocessing techniques to clean and prepare input data.
  - Vectorization methods ensure compatibility with the ML model.
- Machine Learning-Based Prediction
  - Leverages a trained machine learning model to predict possible diseases based on input symptoms.
- Database Integration
  - Fetches cures, medications, or recommendations related to the predicted disease from a pre-populated database.

REAL-WORLD APPLICATIONS

- Preliminary Disease Diagnosis
  - Assists users in identifying potential diseases based on their symptoms before consulting a doctor.
  - Reduces unnecessary visits to healthcare facilities for minor ailments.
- Rural Healthcare Solutions
  - Provides accessible and cost-effective preliminary healthcare support in remote and rural areas where doctors may not be readily available.
- Pandemic Early Detection Systems
  - Can be integrated into public health systems to detect early patterns of symptoms related to pandemics or infectious diseases.



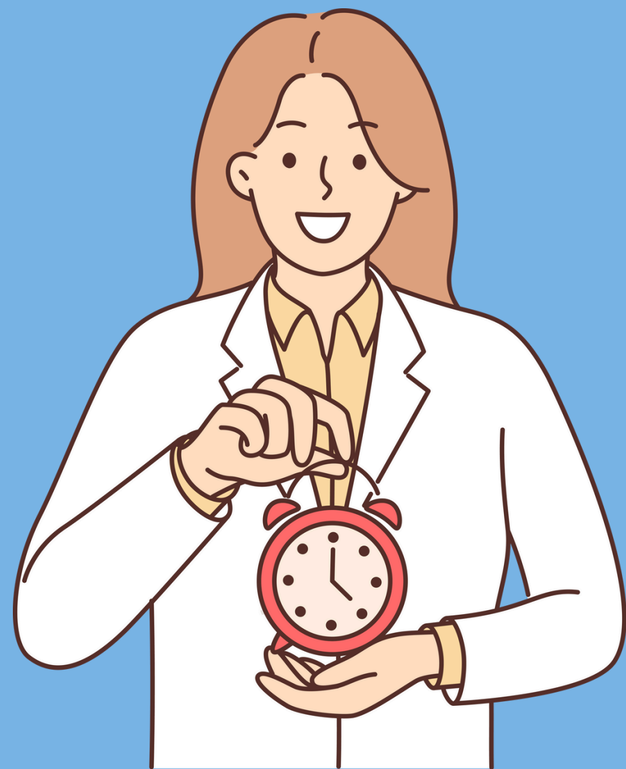
Technical Requirements

Hardware:

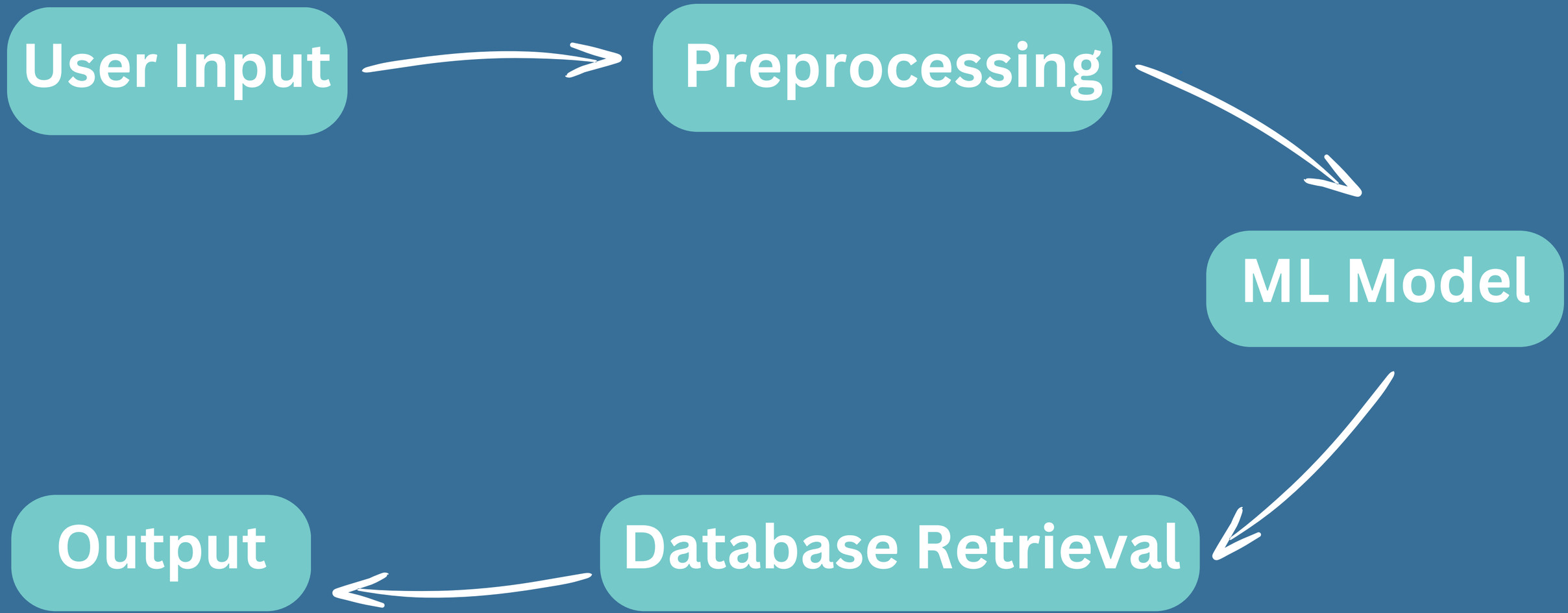
- Processor: Intel Core i5 or higher
- RAM: 8GB or more
- Storage: 1GB for project files and dependencies

Software:

- Python 3.x
- PyQt5 for GUI development
- MySQL for database management
- NLTK, Gensim, and scikit-learn libraries



System Architecture Overview



Literature review

- With advancements in AI and Machine Learning, healthcare systems now utilize predictive models to diagnose diseases early, improving patient outcomes and care.
- Naïve Bayes, Decision Trees, and K-Nearest Neighbors (KNN) are common algorithms used for disease prediction based on structured data like patient symptoms, with Random Forest achieving an accuracy of 98.95%.
- Challenges include symptom overlap, which leads to ambiguities, data privacy issues requiring compliance with standards like GDPR and HIPAA, and the poor generalization of models trained on regional datasets when applied globally.

Future trends involve IoT integration with wearables for real-time monitoring, AI-powered chatbots for personalized prediction, and Natural Language Processing (NLP) to enhance the interpretation of symptoms in multiple languages, improving accessibility.

Module description

The system is modularized into the following key components:

- Login System:
  - Ensures secure authentication by integrating a separate LoginWindow module that verifies credentials without exposing them.
- Database Interaction:
  - Uses MySQL to securely fetch and store disease data and user history, with credentials abstracted in the db_config module.
- Symptom Processing and Disease Prediction:
  - Implements NLP techniques and a trained Word2Vec model to preprocess symptoms and match them to potential diseases using cosine similarity.

Module description

- Voice Input Integration:
  - A threaded system captures voice inputs, converts them to text, and adds them to symptom analysis.
- History Management:
  - Displays user search history with a styled table for a better user experience.
- GUI:
  - Built using PyQt5, with an interactive interface for inputting symptoms, viewing history, and displaying results.

Module Workflow Explanation

- User Login: Access the system using secure credentials.
- Symptom Entry: Input symptoms via text or voice commands.
- Symptom Expansion: Process and expand symptoms using Word2Vec.
- Prediction: Retrieve potential diseases from the database based on symptom similarity.
- Result Display: Show top 5 diseases with additional details.
- History Storage: Save search results for future reference.

Implementation and coding

Programming and Frameworks

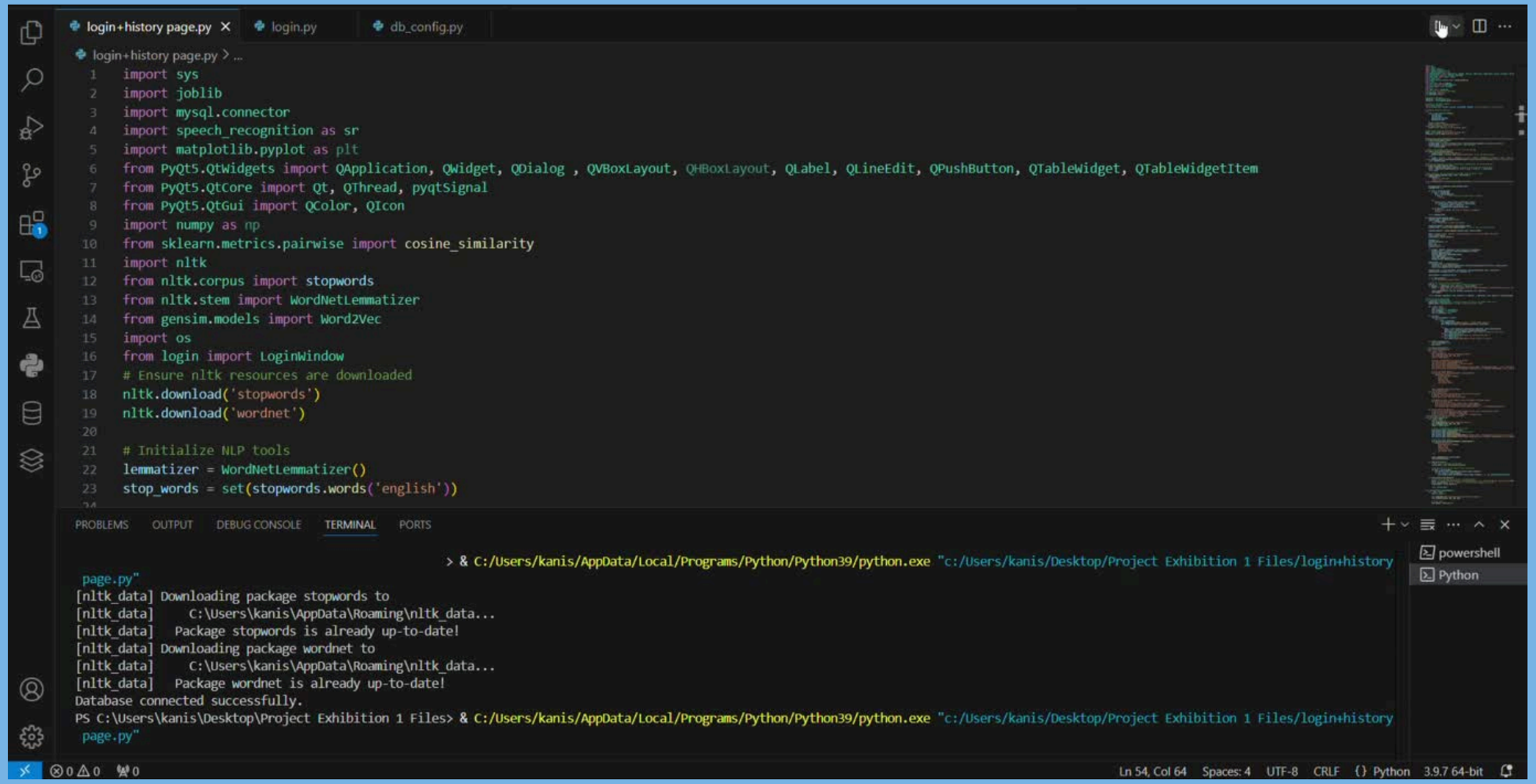
- Programming Language:
  - Python was chosen for its versatility and support for GUI development, database integration, and efficient algorithms.
- GUI Framework:
  - PyQt5 was used to create an interactive and user-friendly graphical interface for symptom input and disease prediction results.
- Database:
  - MySQL-A relational database stores the mapping of symptoms to diseases, along with severity levels.

Implementation and coding

Core Functionality

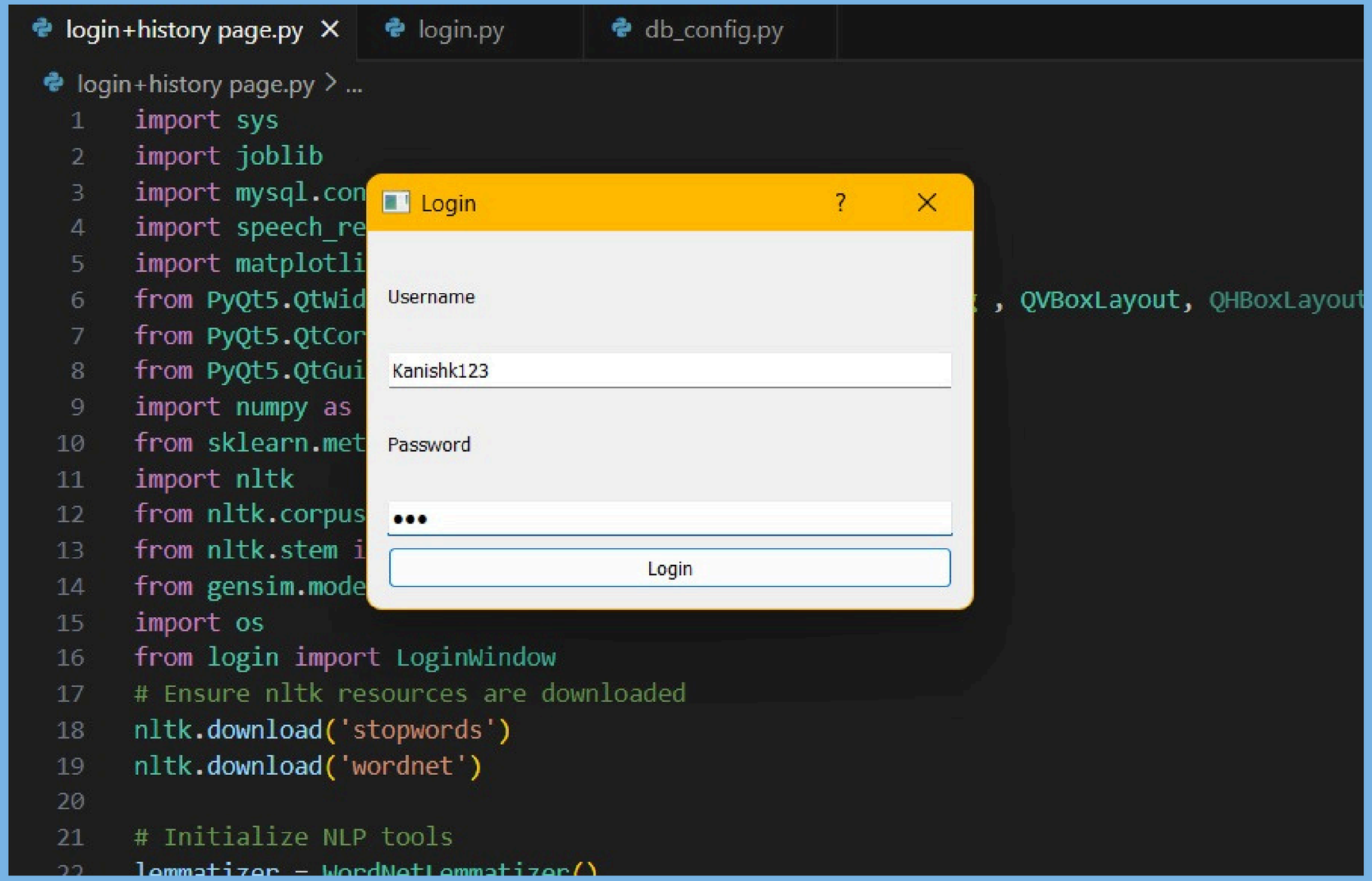
- Symptom Input Module:
  - Allows users to input multiple symptoms via text fields or dropdown options.
  - Validates user input for completeness and relevance.
- Disease Prediction Module:
  - Processes input symptoms using a machine learning model trained on a dataset of symptom-disease mappings to predict potential diseases.
  - Displays a ranked list of potential diseases with severity levels (e.g., mild, moderate, severe).
  - Results are presented in a TreeView table, with severity color-coded for easy interpretation.
- Symptom Suggestion Feature:
  - Automatically suggests additional symptoms related to the input, enhancing prediction accuracy.

Demo Video

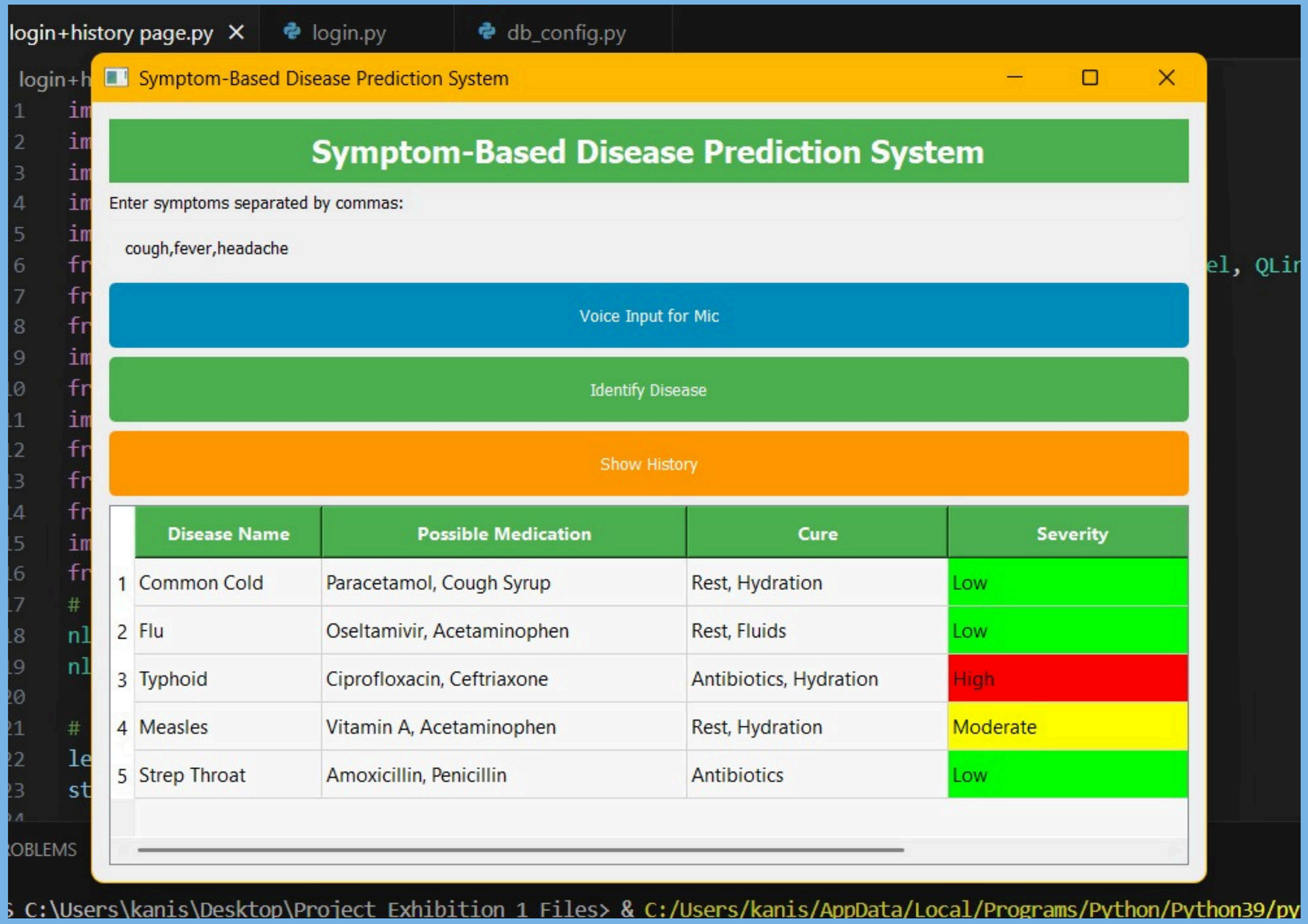

login+history page.py
login.py
db_config.py
login+history page.py > ...
1 import sys
2 import joblib
3 import mysql.connector
4 import speech_recognition as sr
5 import matplotlib.pyplot as plt
6 from PyQt5.QtWidgets import QApplication, QWidget, QDialog , QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QPushButton, QTableWidget, QTableWidgetItem
7 from PyQt5.QtCore import Qt, QThread, pyqtSignal
8 from PyQt5.QtGui import QColor, QIcon
9 import numpy as np
10 from sklearn.metrics.pairwise import cosine_similarity
11 import nltk
12 from nltk.corpus import stopwords
13 from nltk.stem import WordNetLemmatizer
14 from gensim.models import Word2Vec
15 import os
16 from login import LoginWindow
17 # Ensure nltk resources are downloaded
18 nltk.download('stopwords')
19 nltk.download('wordnet')
20
21 # Initialize NLP tools
22 lemmatizer = WordNetLemmatizer()
23 stop_words = set(stopwords.words('english'))
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
> & C:/Users/kanis/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/kanis/Desktop/Project Exhibition 1 Files/login+history page.py"
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\kanis\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data] C:\Users\kanis\AppData\Roaming\nltk_data...
[nltk_data] Package wordnet is already up-to-date!
Database connected successfully.
PS C:\Users\kanis\Desktop\Project Exhibition 1 Files> & C:/Users/kanis/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/kanis/Desktop/Project Exhibition 1 Files/login+history page.py"
powershell
Python
Ln 54, Col 64
Spaces: 4
UTF-8
CRLF
Python
3.9.7 64-bit

Snap shot of project

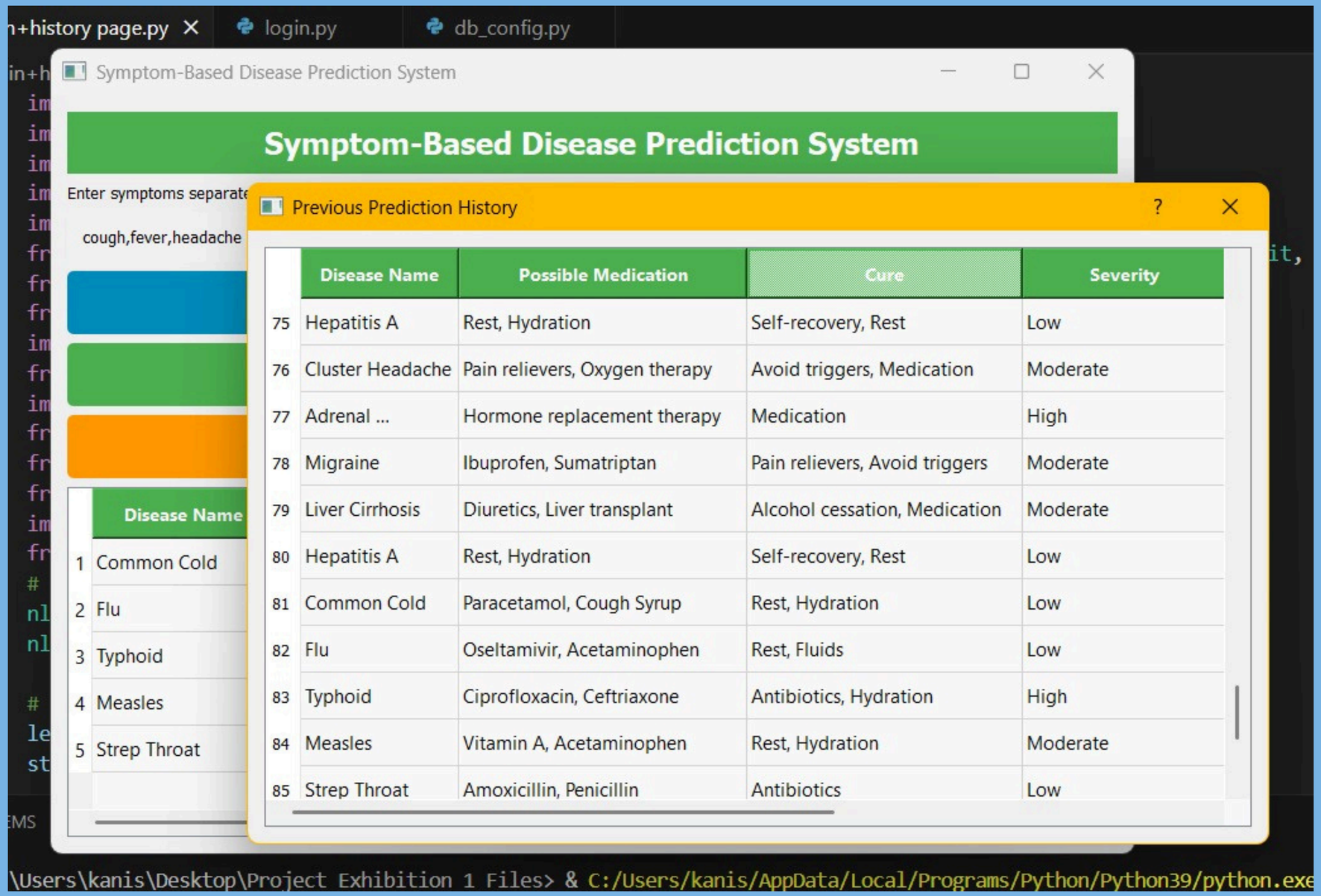
Login page



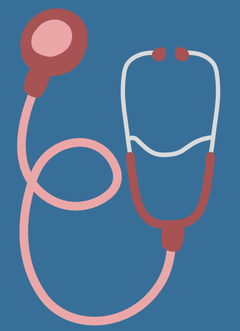
User interface



History Display



Testing



The system was tested in stages:

- Unit Testing: Verified individual components (e.g., speech recognition, machine learning models).
- Integration Testing: Ensured smooth communication between the front end, back end, and database.
- User Testing: Feedback from a small group of users helped refine the system’s usability.

Result and Discussion.

The project successfully integrated machine learning, natural language processing (NLP), speech recognition, and database management to create a fully functioning and user-friendly healthcare application

Conclusion

The Disease Prediction System is an platform that helps users identify potential health conditions based on symptoms. It uses machine learning algorithms, a user-friendly interface, and voice recognition to provide real-time predictions, suggest medications, and track user health trends. The system aims to streamline diagnosis, reduce delays, and enable timely medical attention.


THANK YOU !