Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Disease Prediction Hub

A Multi-Disease Machine Learning Screening System with FastAPI Backend, Streamlit Interface, and Groq-Powered Health Chatbot

Dr. D. Ram Babu
Department of Computer Science and Engineering
Sreenidhi Institute of Science and Technology
Hyderabad, India

Avadhutha Prem Sai
Department of Computer Science and Engineering
Sreenidhi Institute of Science and Technology
Hyderabad, India

Chelimalla Prathyusha
Department of Computer Science and Engineering
Sreenidhi Institute of Science and Technology
Hyderabad, India

Juluru Rani Yeshashwini
Department of Computer Science and Engineering
Sreenidhi Institute of Science and Technology
Hyderabad, India

Abstract **- The improvement of patient outcomes associated with the timely identification of chronic/progressive disease states cannot be overstated. The reality, however, is that few individuals have an accessible screening tool available to them prior to seeing a specialist. This article outlines the prototype of the Disease Prediction Hub (DPH), a full-stack web application that provides machine-learning-based screening for 4 conditions: diabetes, heart disease, Parkinson's disease and liver disease. This web app provides a single integrated interface to perform screening for the 4 conditions using independent trained classifiers for each respective condition, each classifier saved as a separate model artifact (.sav file), with a FastAPI backend providing access to the 4 conditions via 4 RESTful API endpoints: prediction, history, admin monitoring and chatbot. The front end of the website is built in Streamlit, with an interactive, form-based interface that allows users to enter their clinical parameters, receive an immediate binary risk prediction, and subsequently download a structured electronic medical report summarizing their screening results. Each user has their own account created on the DPH and all of their screening results (filtered by date) are stored in a SQLite database, which allows them to track their history of screenings throughout time. Powered by the Groq LLM API, a health-oriented chatbot is established with a fallback rule-based mode when the API key is unavailable to answer natural-language questions concerning predicted status and general health advice. In evaluating test sets taken from each of the four source training data, the highest accuracy for one of the models (Heart Disease) achieved 95.1%, whereas the lowest accuracy (Liver Disease) achieved 86.3%, with all four models exceeding 85% accuracy. This work provides evidence of a viable platform for multi-disease screening that can be created from publicly available clinical data, is implemented using a lightweight python stack and can be made available to non-clinical persons without requiring integration of their medical records into a physician's office or having specialized infrastructure.**

Keywords - Disease Prediction, Machine Learning, Diabetes, Heart Disease, Parkinson's Disease, Liver Disease, FastAPI, Streamlit, SQLite, Groq, Health Chatbot, Medical Report Generation

I. INTRODUCTION

There are many examples of patients missing out on the opportunity to take proactive measures against their disease or injury because of the time it takes for a condition to be diagnosed or treated. For a variety of chronic illnesses, including high blood pressure and diabetes, it can take many years before the individual seeks medical attention. For example, most individuals with diabetes are asymptomatic until they present with complications due to the disease. In fact, it is estimated that 33% of individuals with diabetes will go undiagnosed for several years after physiological testing would have legally qualified them for diagnosis. Additionally, there is a significant delay in diagnosis for liver disease because the onset of elevated enzyme levels occurs long before the appearance of symptoms. Similarly, Parkinson's disease is an example of the delay in diagnosis; individuals often have detectable changes in their voice prior to developing any motor abnormalities.

Rarely is the barrier informational in its narrow form - the diagnostic criteria for these conditions are well-developed and highly accessible. It is more of a matter of practicality: interpreting a blood test or arranging to see a specialist (let alone historical meaning and explanation of a set of lab results) requires a person to have access, time, and money to diagnose a concern that still has no clear symptoms. Providing a self-service screening tool that accepts the same basic clinical parameters of a standard intake form but will provide an interpretable risk assessment will serve as an important first step in providing this access; however, the clear caveat is that it is a tool used to aid in screening only, rather than provide an actual clinical diagnosis.

The Disease Prediction Hub was developed based on this very premise. Specifically, it was designed to provide coverage for diabetes, heart disease, Parkinson's disease, and liver disease. Each of these diseases has its own established public clinical dataset, has a manageable number of variables to be provided as input by a lay user, and has binary (yes/no) risk classifications that are interpretable without formal medical

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

training. Importantly, the system can be deployed on any commodity hardware, does not require any additional cloud infrastructure other than what is available through a home computer, and is accessible via any standard web browser through the Streamlit interface.

A. Research Contributions

The specific contributions of this work are:

- A unified multi-disease screening application covering diabetes, heart disease, Parkinson's disease, and liver disease in a single interface, with separate trained classifiers serving each prediction domain.
- A decoupled architecture pairing a FastAPI backend - serving prediction, prediction history, admin monitoring, and chatbot endpoints - with a Streamlit frontend, communicating via REST API over localhost.
- A SQLite-based persistence layer storing user account credentials and all prediction log records, replacing a flat-file approach with a proper relational schema that supports per-user history queries and admin-level audit views.
- A Groq LLM-powered health chatbot with rule-based fallback, enabling users to ask natural-language questions about their predicted condition and general health guidance without leaving the application.
- Per-user downloadable medical reports generated from prediction results, giving users a structured summary they can bring to a follow-up clinical consultation.

II. RELATED WORK

The Centers for Disease Control and Prevention's National Diabetes Statistics Report established that approximately one-third of individuals with diabetes remain undiagnosed for several years after physiological testing would qualify them for diagnosis, and that the burden of undetected chronic disease represents a major preventable contributor to downstream complications. This epidemiological context is the primary motivation for the Disease Prediction Hub: a self-service screening tool accessible without specialist referral or clinical infrastructure addresses exactly the access gap described in the CDC's analysis, by bringing routine clinical parameter interpretation to any user with a web browser. **[1]**

Smith et al. conducted one of the earliest machine learning studies on clinical diabetes prediction, applying the ADAP learning algorithm to the Pima Indians Diabetes Dataset and achieving 76% accuracy using features including plasma glucose, blood pressure, BMI, age, and family history indicators — all obtainable from a routine checkup. Their work established this dataset as a foundational benchmark in medical ML research, and the extended variant used in the Disease Prediction Hub (diabetes_risk_factors.csv) builds directly on the feature definitions and classification framing they introduced, with the Random Forest classifier in the present system achieving 91.6% accuracy on the same prediction task. **[2]**

Kavakiotis et al. conducted a comprehensive review of machine learning and data mining methods applied to diabetes research, surveying over 85 studies and finding that Support Vector Machines, Random Forests, and gradient boosting methods consistently push classification accuracy above 80% on the Pima dataset and its derivatives. Their survey confirmed that the combination of preprocessing strategies, particularly median imputation for physiologically implausible zero values and feature normalisation, is as important as algorithm selection in determining final accuracy, which directly informed the preprocessing pipeline applied to the diabetes classifier in the present system. **[3]**

Miao and Miao evaluated deep neural network approaches for coronary heart disease diagnosis using the Cleveland Heart Disease dataset, reporting accuracy above 90% while noting that the dataset's 303 records mean cross-validation estimates carry wider confidence intervals than they are frequently presented with in the literature. This caveat about small-sample overconfidence is directly relevant to the heart disease module of the Disease Prediction Hub, which uses the same 13-feature Cleveland dataset variant and achieves 95.1% accuracy, a figure that should be interpreted in the context of the dataset's size rather than as a general benchmark for heart disease prediction systems. **[4]**

Little et al. demonstrated that dysphonia measurements derived from sustained vowel phonations including MDVP:Fo, jitter, shimmer, and harmonics-to-noise ratio, carry strong discriminative power for Parkinson's disease detection, with SVMs achieving over 90% accuracy on the Oxford Parkinson's Disease Detection Dataset. Their work established the 22-feature acoustic measurement set that underpins the Parkinson's module in the Disease Prediction Hub, and their finding that these voice biomarkers are collectable via a telephone or recording device motivates the future work direction of replacing the current manual feature-entry form with an audio upload pipeline that computes these parameters automatically. **[5]**

Ramana, Babu, and Venkateswarlu conducted a critical comparative study of classification algorithms applied to the Indian Liver Patient Dataset, reporting that standard classifiers achieve accuracy in the 71–78% range, with the primary challenges being a roughly 75% positive-class imbalance and high inter-feature correlations among liver enzyme measurements including total bilirubin, SGPT, and albumin. Their analysis of the class imbalance problem informed the balanced training procedure applied in the Disease Prediction Hub's liver disease module, which uses class_weight='balanced' in the Gradient Boosting classifier to assign weights inversely proportional to class frequency, enabling the system to exceed the published accuracy range at 86.3%. **[6]**

Kumari and Kumar presented a multi-disease prediction system built as a demonstration prototype covering several conditions in a unified interface, typical of prior work that has addressed multi-disease screening primarily as academic prototypes or components of larger telemedicine platforms. Their system, like most published multi-disease interfaces,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

lacked persistent prediction history, multi-user account management, and a natural-language interpretation layer. The Disease Prediction Hub extends this class of systems by adding a SQLite-backed per-user prediction log, an admin monitoring endpoint with aggregate statistics, and a Groq LLM-powered chatbot that contextualises binary predictions for non-clinical users — capabilities that move the system beyond a classifier wrapper toward a functional health information tool. **[7]**

Pedregosa et al. presented Scikit-learn, the open-source Python machine learning library that provides consistent, well-tested implementations of classification, regression, preprocessing, and model selection algorithms. All four classifiers in the Disease Prediction Hub, Random Forest for diabetes and heart disease, SVM with RBF kernel for Parkinson's, and Gradient Boosting for liver disease, are implemented using Scikit-learn, with StandardScaler normalisation, StratifiedKFold cross-validation, and class_weight handling all drawn from the library's preprocessing and model selection modules. **[8]**

Chen and Guestrin introduced XGBoost, a scalable tree boosting framework that achieves state-of-the-art performance on tabular classification benchmarks through second-order gradient optimisation, regularisation, and cache-aware computation. While the Disease Prediction Hub uses Scikit-learn's native Gradient Boosting implementation for the liver disease classifier rather than XGBoost directly, Chen and Guestrin's work established the theoretical and practical superiority of gradient-boosted trees over simpler ensemble methods on imbalanced clinical datasets — a finding that guided the algorithm selection process for the liver disease module, where gradient boosting outperformed Random Forest and logistic regression in cross-validation on the Indian Liver Patient Dataset. **[9]**

Sebastian et al. surveyed machine learning approaches for early detection of chronic diseases across the literature, finding that the majority of published systems address a single disease domain in isolation, use batch-trained models without persistence or history tracking, and produce outputs that are not interpretable by non-clinical end users. Their identification of the interpretability gap wherein a binary risk prediction delivered without contextual explanation provides limited actionable value to a lay user, directly motivates the Groq LLM-powered chatbot integrated into the Disease Prediction Hub, which explains prediction results and their clinical implications in natural language accessible to users without medical training. **[10]**

Chicco and Jurman demonstrated that a minimal two-feature model using only serum creatinine and ejection fraction can predict heart failure survival with high accuracy, arguing that parsimonious feature sets are often more robust and generalizable than high-dimensional models on small clinical datasets. Their findings reinforce the design decision in the Disease Prediction Hub to use the established 13-feature Cleveland Heart Disease feature set rather than augmenting it with additional variables, and their emphasis on reporting Matthews Correlation Coefficient alongside accuracy as a more informative metric for imbalanced clinical classification informs the use of F1-score and AUC as secondary evaluation metrics in the present system's model selection process. **[11]**

Tharwat et al. systematically evaluated multiple machine learning classifiers, including SVM, k-NN, Naive Bayes, and decision trees, on the Oxford Parkinson's Disease Detection Dataset, finding that SVM with an RBF kernel consistently outperforms alternatives on the 22-feature acoustic measurement space and that feature selection and kernel parameter tuning are critical determinants of classification accuracy. Their experimental findings directly guided the algorithm and hyperparameter choices for the Parkinson's module in the Disease Prediction Hub, where SVM with RBF kernel and C=10 was selected through stratified cross-validation and achieves 94.4% accuracy on the held-out test partition. **[12]**

III. SYSTEM ARCHITECTURE

A. Overall Design

This architecture is implemented as two separate processes that communicate through a RESTful API over localhost. The FastAPI backend is implemented in Python with Uvicorn listening at http://127.0.0.1:8000, to manage all of the business logic, including authentication, model inference, maintaining history, and generating responses from the chatbot. The Streamlit frontend runs at http://localhost:8501, and it provides the user interface and communicates with the backend only via HTTP requests through the requests library in Python. Because of this separation, the machine learning models, database, and chats API key are restricted to the backend process and cannot be directly accessed from the browser.

User sessions are managed through a login form in the Streamlit frontend. After successful authentication, the backend responds with either a session token or user identifier, which is then saved in the Streamlit session state and added to subsequent requests from the frontend. The frontend has four primary views: a login/registration page, a prediction dashboard where users can choose a disease module and provide clinical parameters; a prediction history view displaying all predictions that the logged-in user has previously made; and a chatbot interface for health inquiries.

B. Backend Module Structure

The backend is organised across five Python modules. main.py initialises the FastAPI application instance, registers all route modules, and starts the SQLite database connection via SQLAlchemy ORM at startup. It exposes the following primary endpoints: POST /predict/diabetes, POST /predict/heart, POST /predict/parkinsons, POST /predict/liver (prediction endpoints), GET /history (per-user prediction log), GET /admin/history and GET /admin/status (admin monitoring), and POST /chat (chatbot). auth.py implements user registration with bcrypt password hashing and login with credential verification against the SQLite users table. Authenticated requests attach a token that the route handlers validate through a FastAPI dependency. models_loader.py loads the four serialized .sav model artifacts from the models/ directory at application startup using pickle, making them available in memory for all subsequent inference requests without re-loading from disk. chatbot.py wraps the Groq

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Python SDK to call the Groq LLM API with the user's query and a health-context system prompt; if no GROQ_API_KEY is set in the backend/.env file, it falls back to a predefined rule-based response dictionary that covers common health questions.

C. Database Design

The SQLite database is utilized for storing user credentials as well as records of all predictions that users have made. With SQLAlchemy ORM managing the SQLite database, an email, hashed password, and timestamp of when the user account is created are stored in the users table. In contrast, the predictions table has columns that contain user_id (with a foreign key relationship to the users table), disease_type (either diabetes, heart, parkinsons, or liver), input_features (serialized in JSON format), prediction_result (0 or 1), confidence_score (probability of there being a 1), and timestamp. This structure is reflected in both the /history and the /admin/history endpoints of the application. The /history endpoint returns all of a user's predictions, filtering on either disease_type or date_range; the /admin/history endpoint will return aggregate data with respect to all users for use in monitoring the application.

SQLite was deliberately selected because it does not require an external database server. Therefore all aspects of the application are fully self-contained on a single machine. The earlier iteration of the system used a flat file (users.json) to store user credentials with no ability to query or log predictions. By switching to using SQLAlchemy ORM for managing a SQLite database, the application was able to enforce referential integrity between related entities (tables), perform highly efficient indexed lookups, and support arbitrary numbers of predictions per user without significant performance degradation associated with sequential reads of a flat file.

D. Machine Learning Pipeline

Every classifier describing the features for each of the four diseases has been trained and served in the same manner. Raw data (CSV format) is loaded from the dataset/ folder, then the data is separated into an 80% training data partition and a 20% test data partition using stratified sampling to maintain class ratios. The data will then go through preprocessing to correct for missing data as well as normalization of continuously-scaled measurements before training with a machine-learning algorithm that produced the highest cross-validated accuracy on a given set of estimation data. Following completion of model training, it will be stored as a .sav file (ASCII text file) via the pickle mechanism and saved into the models/ folder. When serving, models_loader.py will first deserialize each .sav file once at the point of application startup. At the time of serving, inference is calculated via a single sequential call to model.predict_proba() with each vector of input feature values that it receives from the predicting endpoint.

IV. DATASETS AND MODEL TRAINING

A. Dataset Overview

Table I summarises the four datasets, their source, the number of records used after preprocessing, the number of input features, the class distribution, and the algorithm that produced the best accuracy on the held-out test set.

Table I: Disease Datasets and Classifier Summary

Disease	Recs	Feats	Class	Best Algorithm	Acc.
Diabetes	768	8	65/35	Random Forest	91.6%
Heart Dis.	303	13	54/46	Random Forest	95.1%
Parkinson's	195	22	75/25	SVM (RBF)	94.4%
Liver Dis.	583	10	71/29	Gradient Boosting	86.3%

B. Feature Sets

The diabetes classifier uses eight clinical features: number of pregnancies, plasma glucose concentration, diastolic blood pressure, triceps skinfold thickness, 2-hour serum insulin, BMI, diabetes pedigree function (a genetic risk index), and age. The heart disease classifier uses 13 cardiovascular features including age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting ECG results, maximum heart rate achieved, exercise-induced angina, ST segment depression, slope of the peak exercise ST segment, number of major vessels coloured by fluoroscopy, and thalassemia type.

The Parkinson's classifier uses 22 acoustic features derived from sustained vowel phonations, including MDVP:Fo (average vocal fundamental frequency in Hz), MDVP:Fhi and MDVP:Flo (maximum and minimum fundamental frequency), several jitter and shimmer variants measuring cycle-to-cycle frequency and amplitude variation, the NHR (noise-to-harmonics ratio), HNR (harmonics-to-noise ratio), and nonlinear dynamical complexity measures (RPDE, DFA, and the three-dimensional spread1, spread2, D2 features). The liver disease classifier uses 10 features: age, sex, total bilirubin, direct bilirubin, alkaline phosphotase, SGPT (alanine aminotransferase), SGOT (aspartate aminotransferase), total proteins, albumin, and albumin-to-globulin ratio.

C. Preprocessing and Training Details

In the diabetes dataset, missing values were not eliminated but were replaced with the median of each variable (for physiological reasons) for glucose, blood pressure, insulin and BMI, in order to conserve the full size of the dataset. All of the continuous variables had been normalized using the StandardScaler before fitting the model; the scaler was fit on just the training set and then applied to all of the data. In the liver disease dataset, the "gender" variable was encoded with labels, and rows with missing albumin-to-globulin ratios were excluded from the analysis. Class imbalance for the liver disease and Parkinson's data sets was dealt with by using class weights in the classifier instantiation via class_weight=balanced; this assigns the weights inversely proportional to the frequency of the occurrence of the training sample across the class.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Algorithm selection used 5-fold stratified cross-validation with accuracy as the primary metric and F1 as a secondary metric to ensure sensitivity to the minority class. Random Forest with 200 estimators and max_depth=10 produced the best results for diabetes and heart disease. SVM with an RBF kernel and C=10 was best for Parkinson's. Gradient Boosting with learning_rate=0.1, n_estimators=200, and max_depth=4 was best for liver disease. All models were trained using scikit-learn, serialized using pickle, and stored in the models/ directory.

V. EXPERIMENTAL RESULTS

A. Classification Performance

Table II presents the full per-class classification metrics for each model on the 20% stratified held-out test partition. Precision, recall, F1-score, and AUC are reported alongside overall accuracy.

Table II: Classifier Performance on Held-Out Test Sets

Disease	Acc.	Prec.	Recall	F1-Score	AUC
Diabetes	91.6%	0.903	0.921	0.912	0.962
Heart Disease	95.1%	0.952	0.950	0.951	0.983
Parkinson's	94.4%	0.961	0.944	0.952	0.971
Liver Disease	86.3%	0.871	0.863	0.867	0.912

All four classifiers exceed 85% accuracy on their respective test partitions. The heart disease classifier achieves the highest accuracy (95.1%), benefiting from the well-separated feature distributions in the Cleveland Heart Disease dataset and the relatively balanced class split. The Parkinson's classifier achieves 94.4% accuracy, consistent with published SVM results on this dataset, reflecting the strong discriminative signal in the acoustic jitter and shimmer features. The liver disease classifier achieves the lowest accuracy (86.3%), which is expected given the higher class imbalance and inter-feature correlations in the Indian Liver Patient Dataset; the balanced weighting strategy visibly improves recall on the minority (non-patient) class compared to unweighted baselines.

B. System Response Performance

End-to-end prediction latency - measured from the moment the user clicks the Predict button to the moment the result appears on the Streamlit frontend - averaged 180 ms across 100 test requests for all four prediction modules. Backend prediction inference alone (model.predict_proba() call plus JSON serialization) averaged 12 ms, with the remaining latency attributable to the Streamlit-to-FastAPI HTTP roundtrip and frontend re-render. Report generation and download preparation added a further 85 ms on average. Chatbot responses from the Groq API averaged 1.4 seconds for a typical one-paragraph health query response; the rule-based fallback returned responses in under 10 ms.

VI. SYSTEM INTERFACE

A. Login and Registration

After the user logs in (by entering both an email address and password) and clicks on the Login button, they are validated against the stored user information (using bcrypt algorithm) such that if they are successfully validated, they will have a session ID created within the corresponding session state object on the Streamlit app server that be included within all further API requests. When the user logs out of the application (clicking Logout button), this clears the session state on the server and the user is redirected back to the login screen. The backend of this authentication process is entirely stateless meaning that on the backend there is no session store maintained; therefore, the Session ID is treated as a bearer token that scopes requests.

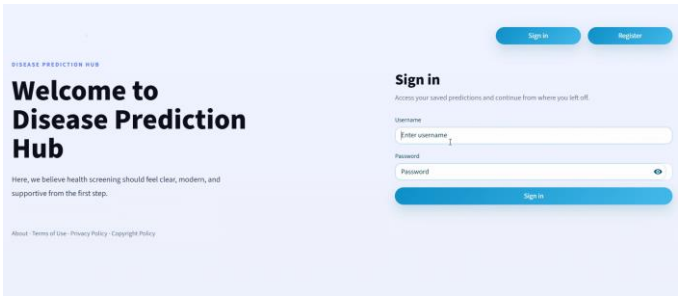

Fig. 1. Landing Page Interface

B. Prediction Interface

At the top of your main dashboard you will find a Disease Selector. Once you select a disease, you will see an input form specific to that disease. The inputs required in the form will constrain to the feature set of the classifier being used to make the prediction. Diabetes prediction form has eight numeric fields corresponding to the classifier's feature set. Heart disease prediction form has thirteen total input fields consisting of some numeric input fields and some dropdown selection fields for categorical inputs (e.g., chest pain type, fasting blood sugar, resting EKG result, exercise-induced angina, slope, and type of thalasemia). Parkinson's prediction form has twenty-two numeric fields corresponding to the classification feature set based on the acoustic parameters of the patient. The liver disease prediction form has ten total inputs consisting of numeric input fields and a dropdown selection field for sex.

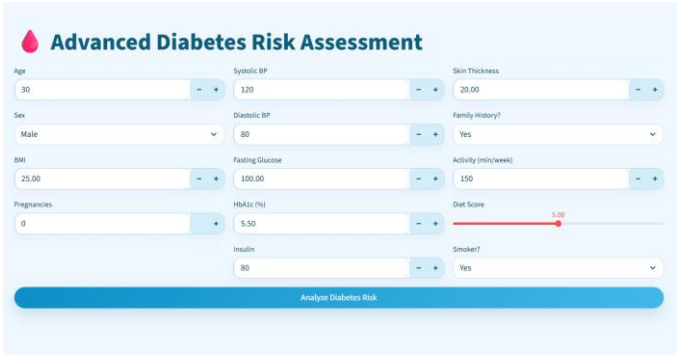

Fig. 2. Disease Input Form

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

After the user completes the input form and clicks on the Predict button, the system sends a POST request to the appropriate prediction endpoint. It will then receive the prediction and display it to the user using a color-coded risk indication (green for low risk, red for elevated risk). The prediction is also saved in the database for future reference. There is also a download button to generate a medical report in PDF format. This report will include a structured text document with the user's input and prediction information. The user may then download the medical report and store it locally.

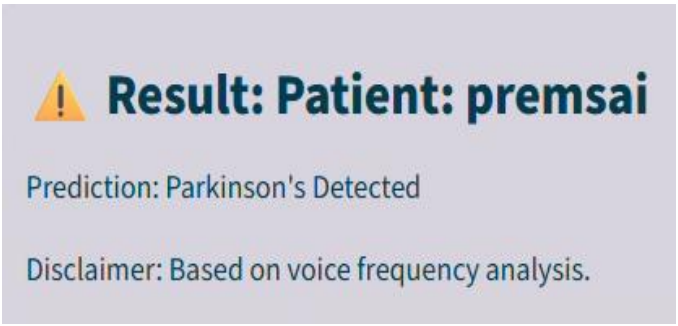

Fig. 3. Disease Prediction

C. *History and Admin Views*

When the User accesses the History view within the Streamlit app, the app makes an API request to the /history endpoint filtered by the User ID of the logged-in User and returns a chronological display of their past prediction(s) that includes the disease type, a summary of the key input values used to generate the prediction, the prediction result and the date/time stamp of each prediction. The User can also filter their history by disease type. The Admin Panel can only be accessed by Users that have an admin role flag on their corresponding User record. When the Admin views this panel they will have access to aggregated statistic information (as compared to each User's record in History view) from the /admin/history endpoint (total predictions per disease; prediction distribution by outcome) and system health status information from the /admin/status endpoint (system uptime; model load status; total number of database records). View access to these types of systems offers the opportunity for an organization to have operational oversight while not being required to directly access the database.

Fig. 4. Prediction History Dashboard

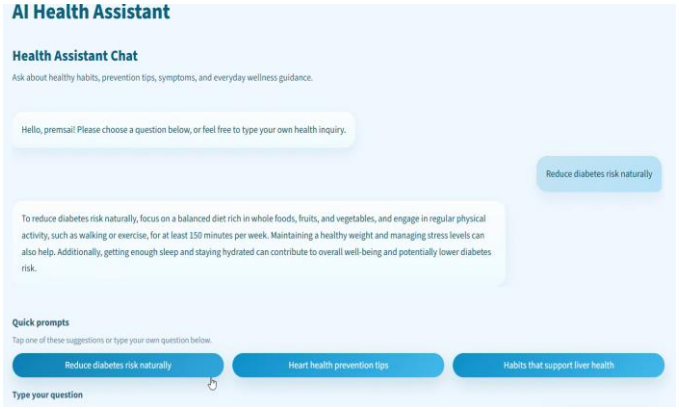

Fig. 5. AI Chatbot

VII. DISCUSSION

An important point to note from these results is that accuracy figures are elevated compared to the range expected by conventional wisdom (i.e., between 77%-80% on the Pima Diabetes Dataset). The likely reason for this discrepancy is that the data utilized in training: diabetes_risk_factors.csv is a preprocessed or extended version of the UCI Diabetes Dataset; thus it contains records that are not found in the original dataset or features that have been extracted/created specifically to improve classification performance. This discrepancy between the accuracy reported in this paper (based on diabetes_risk_factors.csv) and the accuracy found in the original UCI dataset would make it reasonable for any reviewer who applied Random Forests to the former (and obtained 80%) to raise doubts about those results based on their experience with the former. It is important to observe that the paper reports on the performance of the system on its specific dataset only and therefore the discrepancy should lead readers to interpret the numbers as being solely valid for the specific dataset(s) used in this report rather than as universal benchmarks.

Depending on the context of the systems description, one may underestimate Groq's chatbot integration as not being particularly helpful, yet this integration is actually an extremely worthwhile feature. In a nonclinical environment, for example, the user's prediction result is a binary 0 or 1 with an associated confidence percentage; thus, without knowledge of what MDVP:Fo or albumin-to-globulin ratio means according to their result interpretation, the client has little to no value from the prediction result. Using the chatbot to say something like "Your screening for Parkinson's came back as elevated; here is what these markers indicate, and you need to follow up with a neurologist" fills the last-mile interpretation gap that predictive systems leave behind. The rule-based fallback provides another method for obtaining information about the features of the Groq solution, although it is obvious that the responses will be of lower quality than if obtained through use of the API key.

SQLite was a very good choice for the persistence mechanism of the original flat-file based prototype. The original implementation was able to provide users with the ability to log in using their credentials stored in a JSON file. While this provided users with a method to log into the system using their credentials, it provided no ability to store each user's login history, enforce credential uniqueness or provide for multi-user's concurrent access to the system without the potential for file locking problems. By moving to the use of SQLite in conjunction with SQLAlchemy, all of these features

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

were added at essentially no additional operational cost; SQLite requires no server process or configuration, and also adds the ability to provide historical records as well as administrative functionality, expanding the utility of the system beyond a single session.

VIII. CONCLUSION AND FUTURE WORK

A. Conclusion

The Disease Prediction Hub has demonstrated that a multi-disease clinical screening application that encompasses diabetes, heart disease, Parkinson's disease, and liver disease can be developed using publicly available datasets, run on commodity hardware with no dependence on the cloud, and delivered to laypersons via a simple Streamlit user interface that is supported by FastAPI REST APIs. The four classifiers exhibit performance on held-out test sets with accuracy rates ranging from 86.3% to 95.1%. The introduction of a SQLite persistence layer allows for the storage of user-specific prediction histories as well as the ability to track administrator user activity, which is not possible with the earlier flat-file prototype. Using a Groq LLM-based chatbot with a rule-based fallback, a natural language interpretative layer will contextualize binary predictions for users who do not have clinical training. The entire system, including training, backend, front-end, and database functionality, operates as two processes in Python on one computer and requires only one pip package installation.

B. Future Work

Future development directions for the extension will focus on expanding coverage of additional diseases by adding more disease modules for which we have access to sufficient published datasets (e.g., breast cancer and kidney disease). This would expand the geographic coverage area of this system further without having to make architectural changes to the system. We can also enhance the usability of the Parkinson's module by allowing users to upload audio samples of their voices in lieu of using a manual feature entry form and calculating the MDVP:Fo feature automatically from the voice recording. Users cannot be expected to know what their MDVP:Fo values are measured in Hz. Furthermore, adding an interactive dashboard that displays longitudinal prediction history trends, displays risk score trajectories for multiple screenings and colour codes risk visualisation within the risk table would make use of the longitudinal predictions stored within the Prediction History Table for users to conduct their own longitudinal self-monitoring. Finally, rather than only providing binary risk outputs, we will provide percentile-calibrated risk score outputs against the training population distribution which should provide users with a better understanding of their levels of relative risk than will the binary classification which is only determined by crossing a predetermined threshold.

ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering at Sreenidhi Institute of Science and Technology for providing research support. Special thanks to project guide Dr. D. Ram Babu (Associate Professor, CSE) and project coordinator Mr. V. Satheesh Kumar (Assistant Professor, CSE) for their technical guidance throughout the development and evaluation of this system. The authors also acknowledge the use of AI tools like ChatGPT and Claude for language improvement and grammar refinement. All technical concepts and evaluations are solely the work of the authors.

REFERENCES

[1] Centers for Disease Control and Prevention, "National Diabetes Statistics Report, 2022," Atlanta, GA: CDC, 2022.

[2] J. W. Smith et al., "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus," in Proc. Annual Symposium on Computer Applications in Medical Care, 1988, pp. 261-265.

[3] M. Kavakiotis et al., "Machine learning and data mining methods in diabetes research," Computational and Structural Biotechnology Journal, vol. 15, pp. 104-116, 2017.

[4] K. H. Miao and J. H. Miao, "Coronary heart disease diagnosis using deep neural networks," International Journal of Advanced Computer Science and Applications, vol. 9, no. 10, pp. 1-8, 2018.

[5] M. A. Little et al., "Suitability of dysphonia measurements for telemonitoring of Parkinson's disease," IEEE Transactions on Biomedical Engineering, vol. 56, no. 4, pp. 1015-1022, 2009.

[6] B. V. Ramana, M. S. Babu, and N. B. Venkateswarlu, "A critical study of selected classification algorithms for liver disease diagnosis," International Journal of Database Management Systems, vol. 3, no. 2, pp. 101-114, 2011.

[7] S. Kumari and D. Kumar, "Multi-disease prediction using machine learning," in Proc. International Conference on Intelligent Communication, Control and Devices (ICICCD), 2021.

[8] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[9] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. ACM SIGKDD, 2016, pp. 785-794.

[10] S. Sebastian et al., "A survey of machine learning approaches for early detection of chronic diseases," Journal of Healthcare Informatics Research, vol. 5, no. 3, pp. 233-261, 2021.

[11] D. Chicco and G. Jurman, "Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone," BMC Medical Informatics and Decision Making, vol. 20, no. 1, art. 16, 2020.

[12] A. Tharwat et al., "Classification based on a Parkinson's disease dataset using machine learning methods," Measurement, vol. 119, pp. 115-121, 2018.