“ML-Driven Early Detection for Optimal Health – Empowering You with Accurate Predictive Health Analytics”

# SUMMER INTERNSHIP SOPHOMORES PROJECT SPJ 2001­­­­­­

**AP20232410000130**

## Summer Id: 20240007

**First Review**

# UNDER THE GUIDANCE OF

Prof. Anil Vitthalrao Turukmane

**DEPARTMENT OF SCOPE (Speci. Networking And Security)**

# SUBMITTED BY

Madhu Alapaka - 21MIS7022.

Kothamasu Kaarthikeya - 21MIS7039.

Pulluru Nikhilesh - 21MIS7087.

**DEPARTMENT OF SCOPE(Integ M.TechSpec.in Software Engineering)**



**SOFTWARE DESIGN AND ARCHITECTURE**

**(“S.D.A”)**

* **Architectural Representation:**

**Overview:** The system architecture comprises a web-based frontend using Streamlit, a backend for data processing and machine learning model inference, and an optional dataset for user data storage.

* the software architecture pattern that most closely aligns with your setup is the Layered Architecture Pattern. Here’s how it fits your project:
* **Layered Architecture Pattern:**

1. Presentation Layer

Technology: Streamlit

Functionality: Provides a web interface for user interaction.

Components: Input forms for health data, sections for displaying predictions and visualizations.

2. Application Layer

Data Preprocessing Module:

Tasks: Cleaning, normalization, feature engineering.

Libraries: Pandas, NumPy

Model Training and Validation Module:

Tasks: Training machine learning models, validating performance.

Libraries: Scikit-learn, TensorFlow/Keras

Models: Logistic Regression, Random Forest, SVM, Neural Networks

Evaluation: Accuracy, precision, recall, F1-score, ROC-AUC.

Prediction Module:

Tasks: Real-time predictions based on user input.

Libraries: Trained models (Scikit-learn, TensorFlow/Keras)

Integration: Embedded within the Streamlit app.

Reporting Module:

Tasks: Generate performance reports and visualizations.

Libraries: Matplotlib, Seaborn

3. Data Layer

Data Sources:

Datasets: Diabetes, Heart Disease, Parkinson's Disease datasets from reliable sources.

Storage: Local files or cloud storage.

4. Infrastructure Layer

Development Environment:

Tools: Jupyter Notebook, VS Code

Dependencies: Virtual environment setup with necessary libraries.

Deployment Environment:

Platform: Local machine or cloud-based platforms (AWS, Google Cloud, Azure).

Configuration: High-performance laptops with potential GPU support.

Explanation of Fit

Layered Structure: Your project clearly separates concerns into different layers (Presentation, Application, Data, Infrastructure).

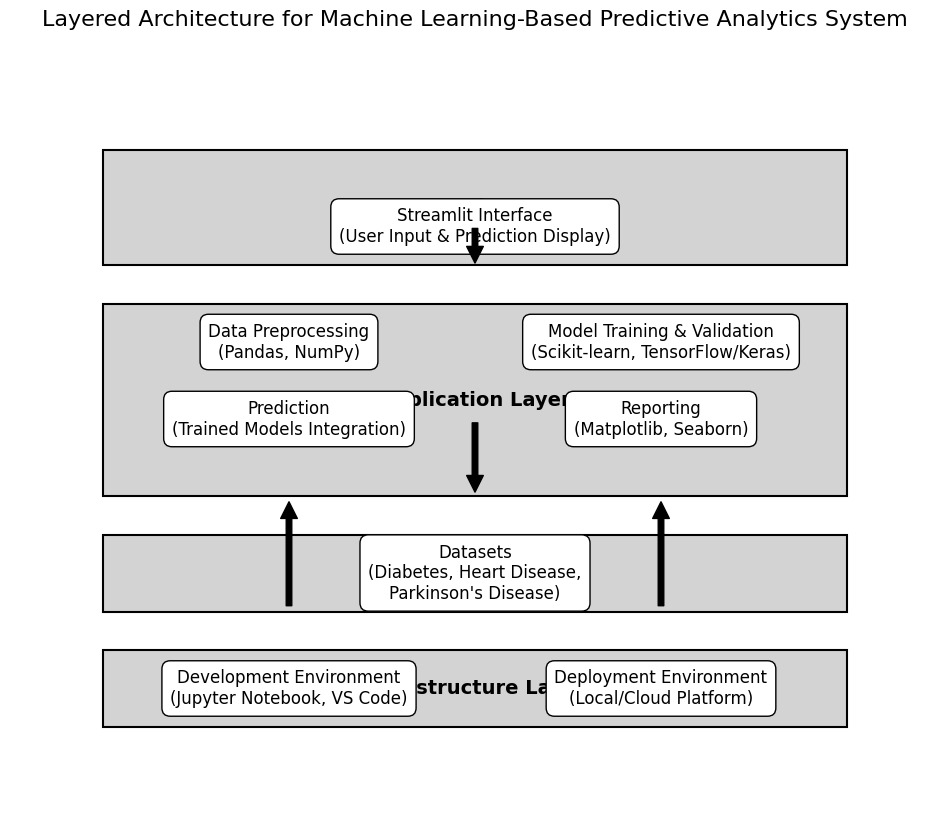
Modularity: Each module has a distinct responsibility, promoting a clean separation of concerns.

Scalability and Maintenance: The layered architecture makes it easier to maintain and scale individual components of the application.

User Interface: The Presentation Layer is distinct and uses Streamlit for the front-end, aligning well with the layered pattern.

**Conclusion**

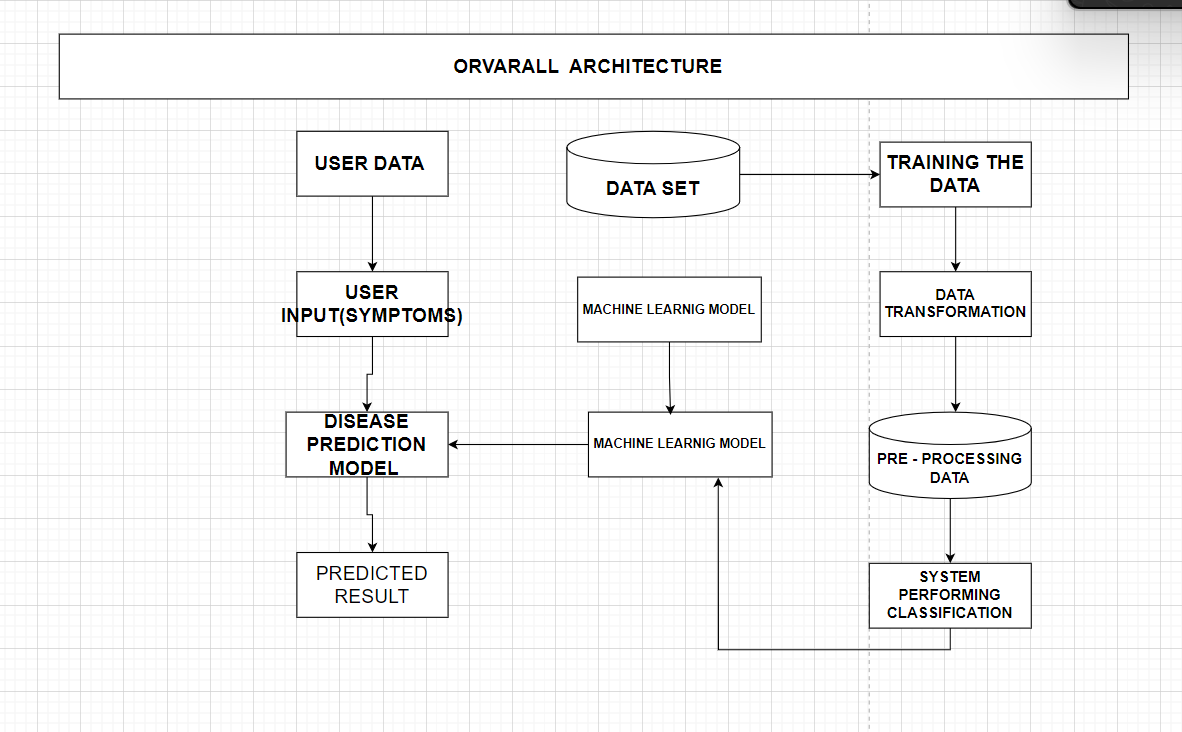
While other architecture patterns could potentially be used depending on further specifics and scalability requirements. It emphasizes the separation of concerns, modularity, and a clear hierarchical structure, making it suitable for developing and maintaining a complex system like a multiple disease prediction application.



* **THE MAJORLY THE SOFTWARE DESIGN AND ARCHITECTURE (“S.D.A”) :**
* Software Architecture & Design - The architecture of a system describes its major components, their relationships (structures), and how they interact with each other.
* Software architecture and design includes several contributory factors such as Business strategy, quality attributes, human dynamics, design, and IT environment.
* Also, we are specified that S.D.A as majorly 3 designs and architecture

**I.OVERALL ARCHITECTURE DIAGRAM:**

* An architectural diagram is a visual representation that maps out the physical implementation of components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.



**Detailed Explanation of the Overall Architecture Diagram:**

* The architecture diagram is the workflow of the MLDriven Early Detection for Optimal Health system. It can be divided into several main components, each playing a critical role in the disease prediction process.

**Steps:**

1. User Data Input:

User Input (Symptoms): The process starts with the user entering their symptoms or health data into the system. This data is crucial as it serves as the primary input for the prediction model.

2. Data Set and Machine Learning Model Training:

Data Set: This block represents the historical medical data collected from various sources. This data is used to train the machine learning models.

Training the Data: This step involves preparing the data for training. It includes:

Data Transformation: Converting the raw data into a suitable format for machine learning algorithms.

Preprocessing Data: Cleaning and normalizing the data to ensure high quality and consistency.

System Performing Classification: Using various algorithms to classify the data into different disease categories.

Machine Learning Model: After preprocessing, the data is used to train machine learning models. These models learn patterns and relationships within the data, which are crucial for accurate predictions.

3. Disease Prediction Model:

Machine Learning Model: The trained machine learning models are applied to the user input data. These models analyze the symptoms provided by the user and predict the likelihood of different diseases.

Disease Prediction Model: This is the core component where the actual prediction happens. The machine learning model processes the user input data and generates a predicted result.

4. Predicted Result:

Predicted Result: The final output of the system is the predicted result. This result indicates the likelihood of the user having a particular disease based on the input symptoms.

**Conclusion**

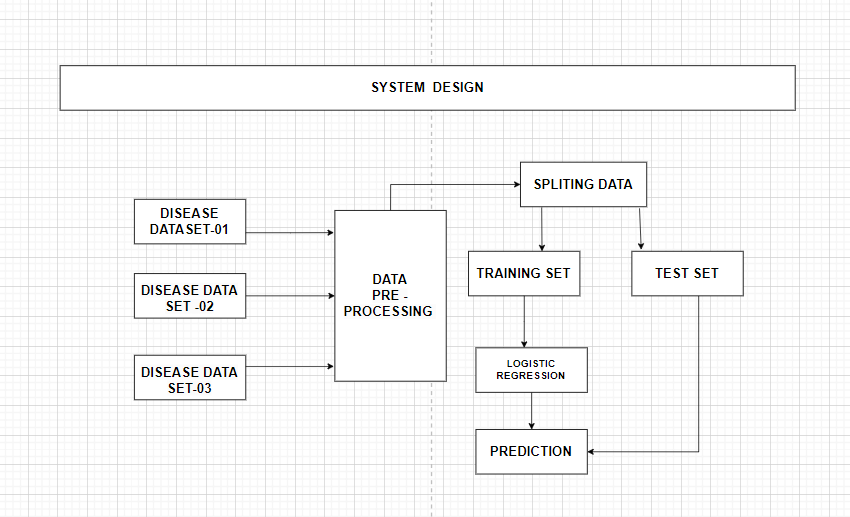
* The architecture of the MLDriven Early Detection for Optimal Health Systems demonstrates a robust framework for disease prediction using machine learning. By integrating user input with historical data, transforming and preprocessing the data, and utilizing advanced machine learning models, the system can provide accurate and timely predictions for various diseases.

1. User-Friendly: The system is designed to be user-friendly, allowing individuals to input their symptoms easily and receive quick predictions.
2. DataDriven: Leveraging a vast dataset and machine learning, the system ensures high accuracy and reliability in predictions.
3. Comprehensive Workflow: The workflow from data input to prediction result is well-structured, ensuring that each step is optimized for the best performance.
4. Scalable: The architecture is scalable, allowing for the integration of additional diseases and improvements in prediction models as more data becomes available.

In conclusion, this architecture lays a solid foundation for a powerful disease prediction system, aiming to revolutionize how early detection and preventive healthcare are approached. By continuing to refine and expand this system, it can significantly contribute to better health outcomes and proactive disease management.

**II.SYSTEM DESIGN:**

* Systems Design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It involves translating user requirements into a detailed blueprint that guides the implementation phase.



**Detailed Explanation of the System Design Diagram:**

* The system design diagram provides a step-by-step workflow of the disease prediction model, from data collection to prediction. Below is a detailed explanation of each step in the process:

**Steps:**

1. Disease Datasets:

Disease Dataset01, Disease Dataset02, Disease Dataset03: These datasets represent various sources of disease-related data. Each dataset contains records of patient symptoms, medical history, and diagnosis information.

2. Data PreProcessing:

Data PreProcessing: This is a crucial step where the raw data from different datasets is cleaned and transformed to ensure quality and consistency. It involves handling missing values, normalizing data, and encoding categorical variables.

3. Splitting Data:

Splitting Data: After preprocessing, the data is split into two main sets:

Training Set: This set is used to train the machine learning model. It constitutes the majority of the data.

Test Set: This set is used to evaluate the performance of the trained model. It helps in assessing how well the model generalizes to new, unseen data.

4. Training Set:

Training Set: The subset of the data used to train the machine learning algorithms.

5. Logistic Regression:

Logistic Regression: This specific algorithm is used for training in the given system design. Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. It is particularly useful for binary classification problems (e.g., predicting the presence or absence of a disease).

6. Prediction:

Prediction: Once the logistic regression model is trained, it is used to make predictions on new data. The model analyzes the test set and provides predictions based on the learned patterns.

7. Test Set:

Test Set: This subset of the data is used to test the performance of the trained model. The predictions made on the test set are compared with the actual outcomes to evaluate the accuracy and reliability of the model.

**Conclusion**

* The system design for the MLDriven Early Detection for Optimal Health system illustrates a comprehensive and systematic approach to developing a disease prediction model. Here are the key points:

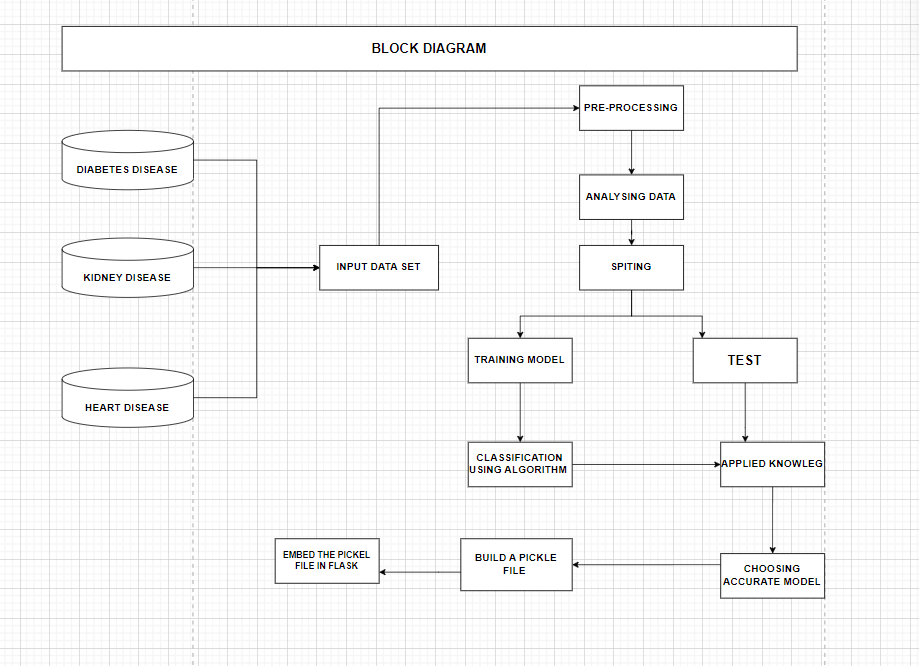
1. Data Integration and PreProcessing: The integration of multiple disease datasets and thorough preprocessing ensure that the data fed into the model is clean, consistent, and suitable for analysis.

1. Model Training and Evaluation: Splitting the data into training and test sets allows for effective training of the logistic regression model and accurate evaluation of its performance.
2. Logistic Regression for Prediction: Utilizing logistic regression, a robust and interpretable machine learning algorithm, enhances the model's ability to predict the likelihood of diseases based on user input.
3. Reliable Predictions: By rigorously testing the model on unseen data, the system ensures that the predictions are reliable and generalizable to new cases.

In summary, this system design lays out a clear and efficient workflow for building a disease prediction model. By following these steps, the system aims to provide accurate and timely predictions, contributing significantly to early disease detection and proactive healthcare management.

**III. BLOCK DIAGRAM :**

* A block diagram is a drawing illustration of a system whose major parts or components are represented by blocks.



**Detailed Explanation of the System Design Diagram**

* The system design diagram provides an in-depth view of the process flow from data acquisition to the final prediction. Below is a detailed step-by-step explanation of each component in the diagram:

**Steps:**

1. Disease Data Set01, Disease Data Set02, Disease Data Set03:

These are the different datasets containing disease-related data. Each dataset might correspond to different diseases such as diabetes, heart disease, and kidney disease.

2. Data PreProcessing:

Data PreProcessing: This is a critical step where raw data from the datasets is cleaned and prepared for analysis. This includes:

Data Cleaning: Handling missing values, removing duplicates, and correcting inconsistencies.

Data Transformation: Normalizing data, encoding categorical variables, and scaling numerical features.

Feature Engineering: Creating new features or modifying existing ones to improve model performance.

3. Splitting Data:

After preprocessing, the data is split into training and test sets. This step is essential for evaluating the model's performance on unseen data.

Training Set: This subset of the data is used to train the machine learning models.

Test Set: This subset is used to evaluate the performance of the trained model and ensure it generalizes well to new data.

4. Training Set:

Training Set: The training set is used to fit the machine learning model. The model learns patterns and relationships within the data during this phase.

5. Logistic Regression:

Logistic Regression: This is one of the machine learning algorithms used for training. It is particularly effective for binary classification problems, such as predicting the presence or absence of a disease.

6. Prediction:

Prediction: Once the model is trained, it can be used to make predictions on new, unseen data. The predictions indicate the likelihood of the disease based on the input features.

**Conclusion**

The system design diagram provides a comprehensive view of the data flow and processing involved in the disease prediction system. Key points include:

1. Diverse Data Sources: The system uses multiple datasets, each potentially representing different diseases, to ensure a broad and inclusive model training process.
2. Robust Data PreProcessing: Critical steps are taken to clean, transform, and enhance the data, ensuring that the models are trained on high quality and relevant features.
3. Effective Data Splitting: The data is split into training and test sets to train the model effectively and evaluate its performance rigorously.
4. Utilization of Logistic Regression: Logistic regression is used as one of the primary algorithms, suitable for binary classification tasks in disease prediction.
5. Accurate Predictions: The trained model can predict the likelihood of diseases based on new user data, aiding in early detection and intervention.

* In summary, the system design diagram outlines a structured and systematic approach to developing a machine learning-based disease prediction system. By following these steps, the system aims to provide accurate and timely predictions, supporting proactive healthcare management and better health outcomes.
* **IMPLEMENTATION:**
* Multiple Disease Prediction System using Machine Learning in Python | Streamlit Web App - Deployment" involves creating a web application for predicting multiple diseases using machine learning algorithms. The framework and tools used in the project include:

1. **Machine Learning Algorithms**: Logistic Regression, Decision Tree, Random Forest, etc., for disease prediction.
2. **Python Libraries**: pandas, scikit-learn, numpy, etc., for data manipulation and model training.
3. **Streamlit**: For building and deploying the web application.
4. **Deployment**: The application is deployed using Streamlit sharing or similar services

* Jupyter notebook "HeartExploration.ipynb" primarily focuses on predicting heart disease using the Support Vector Machine (SVM) algorithm. Here's a detailed breakdown of the algorithms and framework used:

### **Algorithms:**

* **Support Vector Machine (SVM)**: The notebook uses svm.SVC with a linear kernel to build the classification model for predicting heart disease.

### **Framework and Tools:**

* **Python Libraries**:
  + **NumPy**: For numerical operations.
  + **Pandas**: For data manipulation and analysis.
  + **Seaborn**: For data visualization.
  + **Matplotlib**: For plotting graphs.
  + **scikit-learn**: For machine learning tasks including model training and evaluation.
  + **Pickle**: For saving and loading the trained model.

### **Workflow:**

1. **Data Loading**: The dataset is loaded using pandas.
2. **Data Exploration**: Initial exploration includes checking the shape, description, and missing values in the dataset.
3. **Data Visualization**: Pair plots and group-by operations to understand data distribution and relationships.
4. **Data Preprocessing**: Separating features (X) and target (Y), and splitting the data into training and test sets.
5. **Model Training**: Using svm.SVC with a linear kernel to train the model on the training data.
6. **Model Evaluation**: Predicting on the test set and calculating the accuracy.
7. **Model Deployment**: Using Pickle to save the trained model and demonstrate how to load and use the model for predictions.

### **Component Design:**

#### **Frontend (Streamlit)**

The frontend of the Multiple Disease Prediction System is built using Streamlit, a Python library for creating interactive web applications. The frontend is responsible for providing an intuitive and user-friendly interface where users can input their health data and view prediction results.

**Key Components:**

1. **Input Forms for Collecting User Health Data:**
   * **User Interface Design:**
     + The interface will be designed with clear labels and instructions to guide users through the data input process.
     + Input fields will be organized logically, often grouped by type (e.g., personal information, vital signs, blood test results).
   * **Input Fields:**
     + **Personal Information:**
       - Age, gender, weight, height
     + **Vital Signs:**
       - Blood pressure, heart rate, cholesterol levels
     + **Lifestyle Factors:**
       - Smoking status, exercise frequency, diet habits
     + **Specific Health Metrics:**
       - Glucose levels for Diabetes prediction
       - Chest pain type, maximum heart rate achieved for Heart Disease prediction
       - Motor skills, speech patterns for Parkinson’s Disease prediction
   * **Form Validation:**
     + Ensuring required fields are filled.
     + Validating input types (e.g., numbers for age, categorical data for gender).
2. **Display Areas for Showing Prediction Results:**
   * **Prediction Results Section:**
     + Once the user submits their health data, the results will be displayed in a dedicated section.
     + The results section will include:
       - Predicted risk levels (e.g., low, medium, high)
       - Detailed breakdown of risk factors
       - Visual aids such as graphs or charts to illustrate the results.
   * **Health Recommendations:**
     + Based on the prediction results, the system may provide general health recommendations.
     + Suggestions for lifestyle changes or further medical consultation.
   * **User Feedback:**
     + Option for users to provide feedback on the predictions and the system’s usability.
     + Collecting user feedback for continuous improvement.

#### **Backend (Python)**

The backend of the system is responsible for processing the input data, running the machine learning models, and communicating the results back to the frontend. It is built using Python, leveraging various libraries and frameworks for data processing and machine learning.

**Key Components:**

1. **Data Preprocessing Module:**
   * **Transformation:**
     + Scaling features to a uniform range (e.g., using MinMaxScaler or StandardScaler from scikit-learn).
     + Encoding categorical variables (e.g., one-hot encoding for gender).
     + Feature selection and extraction to improve model performance.
2. **Machine Learning Models:**
   * **Model Training:**
     + Training separate models for each disease using relevant datasets.
     + Using techniques such as cross-validation and hyperparameter tuning to optimize model performance.
   * **Algorithms:**
     + **Diabetes Prediction:**
       - Logistic Regression, Random Forest, Gradient Boosting
     + **Heart Disease Prediction:**
       - Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Neural Networks
     + **Parkinson’s Disease Prediction:**
       - Random Forest, XGBoost, Decision Trees
   * **Model Evaluation:**
     + Evaluating model performance using metrics such as accuracy, precision, recall, and F1-score.
     + Ensuring models are not overfitting and can generalize well to new data.
3. **APIs for Communication Between Frontend and Backend:**
   * **RESTful APIs:**
     + Developing RESTful APIs using frameworks like Flask or FastAPI.
     + Endpoints for:
       - Submitting user data (/submit\_data)
       - Retrieving prediction results (/get\_results)
       - User authentication and management (if applicable)
   * **Data Handling:**
     + Ensuring secure transmission of user data between frontend and backend.
     + Using JSON format for data exchange.

* **EXPERIMENT ALGORITHM PERFORMING FIRST IMPLEMENTATION OF PROGRAMMING CODE :**

* 1. **Background:**
* In India, huge mortality occurs due to cardiovascular diseases (CVDs) as these diseases are not diagnosed in early stages. Machine learning (ML) algorithms can be used to build efficient and economical prediction system for early diagnosis of CVDs in India.
  1. **Methods**
* A total of 1670 anonymized medical records were collected from a tertiary hospital in South India. Seventy percent of the collected data were used to train the prediction system. Five state-of-the-art ML algorithms (k-Nearest Neighbours, Naïve Bayes, Logistic Regression, AdaBoost and Random Forest [RF]) were applied using Python programming language to develop the prediction system. The performance was evaluated over remaining 30% of data. The prediction system was later deployed in the cloud for easy accessibility via the Internet.
  1. **Results**
* ML effectively predicted the risk of heart disease. The best performing (RF) prediction system correctly classified 470 out of 501 medical records thus attaining a diagnostic accuracy of 93.8%. Sensitivity and specificity were observed to be 92.8% and 94.6%, respectively. The prediction system attained positive predictive value of 94% and negative predictive value of 93.6%.
  1. **Conclusions**
* ML-based prediction system developed in this study performs well in early diagnosis of CVDs and can be accessed via Internet. This study offers promising results suggesting potential use of ML-based heart disease prediction system as a screening tool to diagnose heart diseases in primary healthcare centres in India, which would otherwise get undetected.

**5.Summary:**

* The system is designed to provide a robust, scalable, and user-friendly platform for early disease detection using machine learning. The architecture ensures modularity, allowing for easy updates and maintenance, while the use of Streamlit ensures an intuitive user experience.