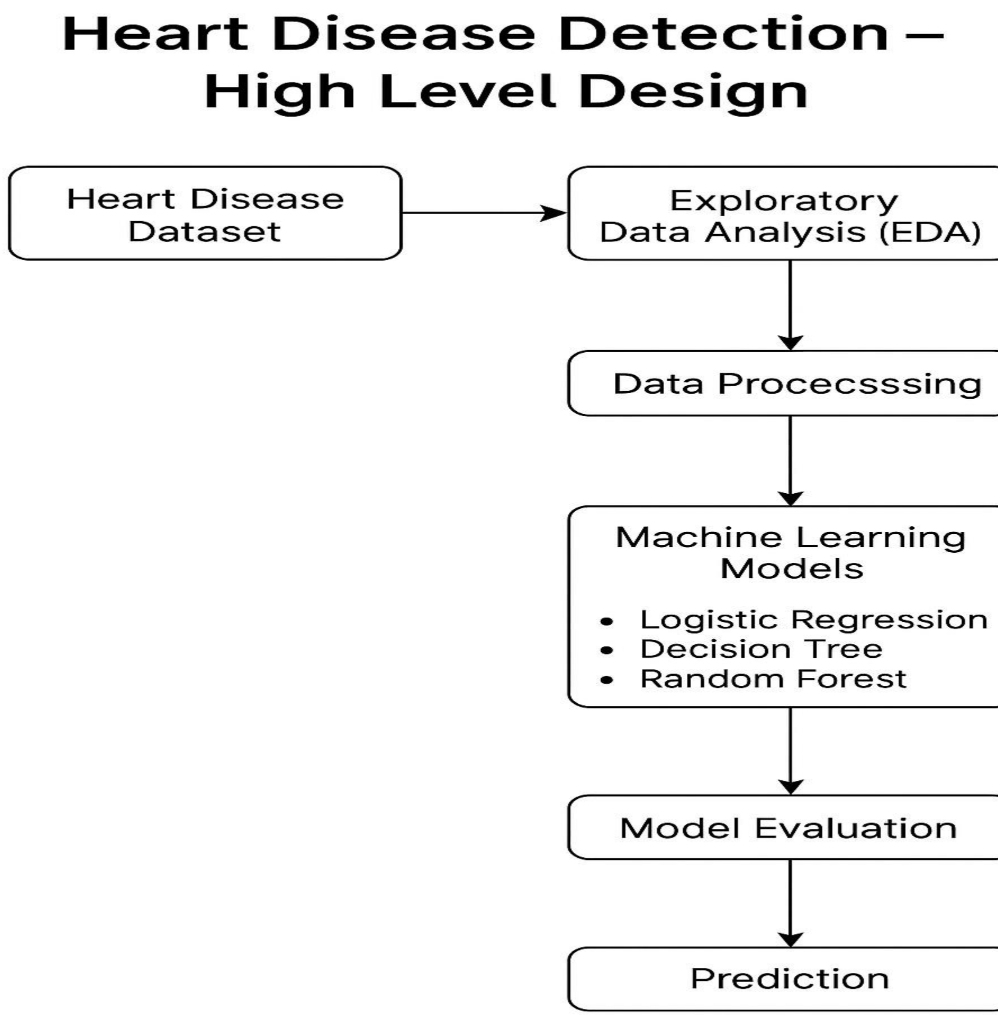
Heart Disease Prediction Using ML

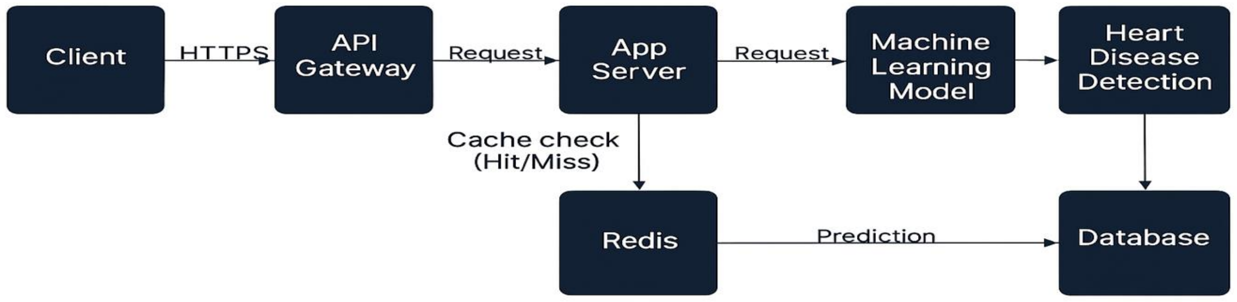
Heart disease remains one of the leading causes of death worldwide, affecting millions of individuals each year. Early diagnosis and timely treatment are critical for reducing mortality and improving the quality of life for patients. However, traditional diagnostic methods can be time-consuming, subjective, and dependent on expert interpretation.

In recent years, machine learning (ML) has emerged as a powerful tool in the healthcare sector. By analyzing large volumes of clinical data, ML algorithms can uncover hidden patterns and make accurate predictions that assist healthcare professionals in decision-making.

This project focuses on leveraging machine learning techniques to predict the presence of heart disease based on patient data. Using a publicly available dataset, the system is trained to classify whether a patient is likely to have heart disease based on clinical attributes such as age, chest pain type, cholesterol levels, and other vital indicators.



High Level Diagram (HLD):



Detailed Explanation of the Architecture

1. Client

- **Role:** The end user interface – typically a web or mobile application.
- **Action:** Sends **input features** (like age, cholesterol, etc.) via an HTTPS request to the system.
- **Security:** Data is transmitted securely using HTTPS.

2. API Gateway (with Rate Limiter)

- **Role:** The first entry point into your backend system.
- **Functions:**
  - Accepts HTTPS requests from the client.
  - Applies **rate limiting** to prevent abuse (e.g., too many requests from a single user/IP).
  - Forwards valid requests to the application server.

3. Application Server

- **Role:** Core backend logic handler (usually built with Flask/FastAPI/Django).

- **Functions:**
  - **Authenticates** the request (if needed).
  - Performs **basic validation and preprocessing**.
  - Initiates a **cache check** with Redis (see below).
  - If prediction isn't cached, forwards request to the ML model service.

4. Redis (Caching Layer)

- **Role:** Speed up performance by caching previous predictions.
- **Flow:**
  - App Server queries Redis first for a **"cache hit/miss"** using the input features as the key.
  - If **hit:** Returns cached prediction directly (saves compute time).
  - If **miss:** Proceeds to the ML model and stores the result in Redis for future use.

5. Machine Learning Model

- **Role:** Core prediction engine.
- **Functions:**
  - Receives preprocessed input from the App Server.
  - Runs the input through a trained model (e.g., Random Forest, Logistic Regression).
  - Returns a prediction (0 = No Disease, 1 = heart disease).
- **Deployment:** Often containerized and deployed as a separate microservice (via Flask API, TensorFlow Serving, etc.).

6. Heart Disease Detection Module

- **Role:** Business logic component that interprets and formats the raw prediction.
- **Example Output:**
  - "The patient is at risk of heart disease" (for 1)
  - "No signs of heart disease detected" (for 0)

7. Database

- **Role:** Stores permanent logs of:
  - Predictions
  - Input features
  - Timestamps
  - User metadata (if applicable)
- **Use Cases:**
  - Audit history
  - Retraining the model on new data
  - Generating reports/analytics

Conclusion

This HLD outlines a clean, scalable, and efficient architecture to support heart disease prediction using machine learning. The separation of services, usage of caching, and optimized prediction workflow ensures high performance and reliability in a production-ready system.

Flow Summary

1. User sends input → API Gateway → App Server
2. App Server checks Redis for cached result
3. If not cached, forwards to ML Model
4. ML Model generates prediction → passed to Heart Disease Detection
5. Result is saved to DB and/or Redis → returned to client.