



Introduction




Designing an Expert system for
diagnosing Heart Disease 1.0





Team Members



- Navyaprabha Rajappa
 - Chaitanya Nalluru
 - Atharva Atre
 - Sai Vikram Gurram
- 

Agenda

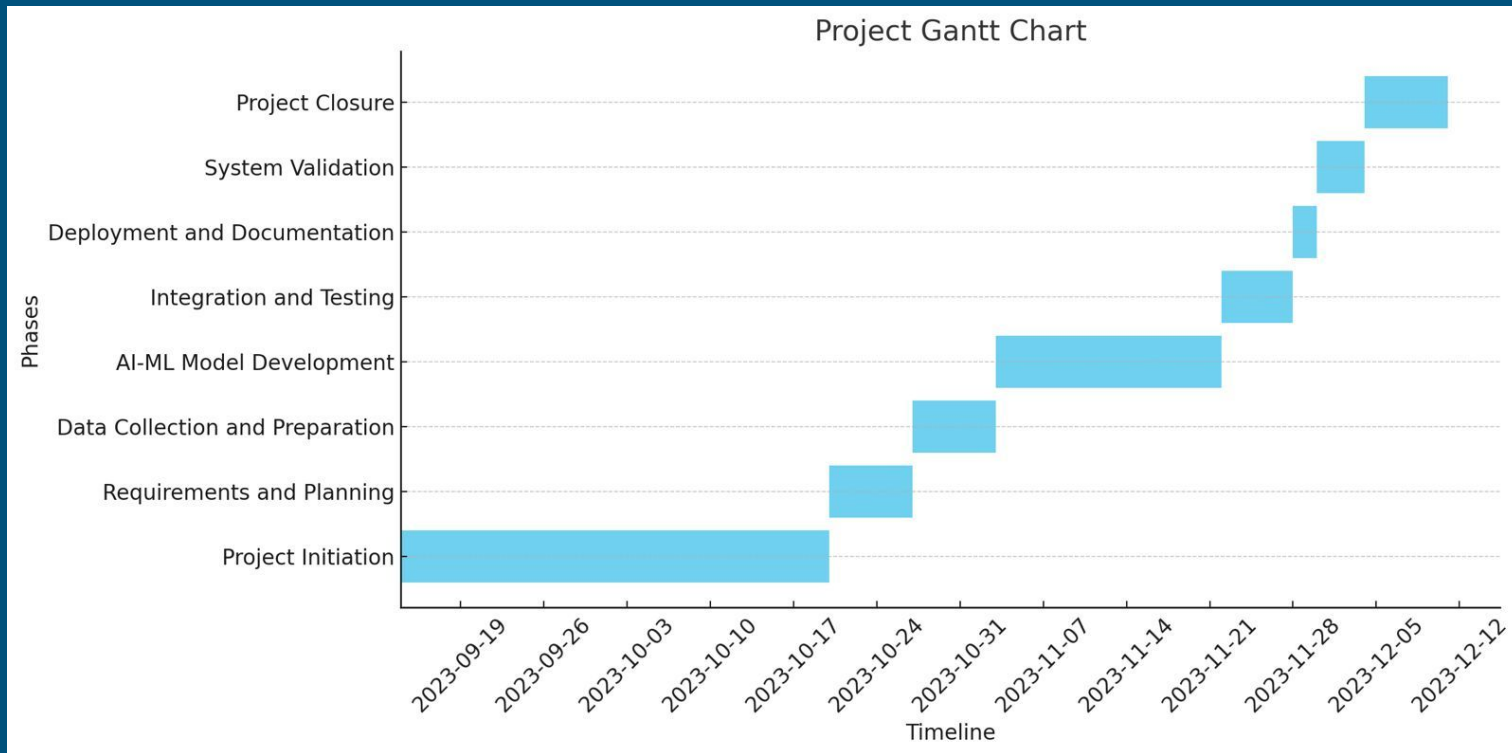
- Team Organization and Process
- Sprint Report
- Use Case Diagram, Sequence Diagram, Class Diagram, State Diagram
- About Model
- Front End
- Back End and integration
- Tools and development Environment
- Project Demo

Team Organization and Process

- Phase 1: Project Initiation(By September 14th)
- Phase 2: Requirements and Planning(By October 20th)
- Phase 3: Data Collection and Preparation(By October 27th)
- Phase 4: AI-ML Model Development(By November 3rd)
- Phase 5: Integration and Testing(By November 26th)
- Phase 6: Deployment and documentation(by November 28th)
- Phase 7: System Validation(November 30th)
- Phase 8: Project closure(December 4th)

Timeline

Chart:



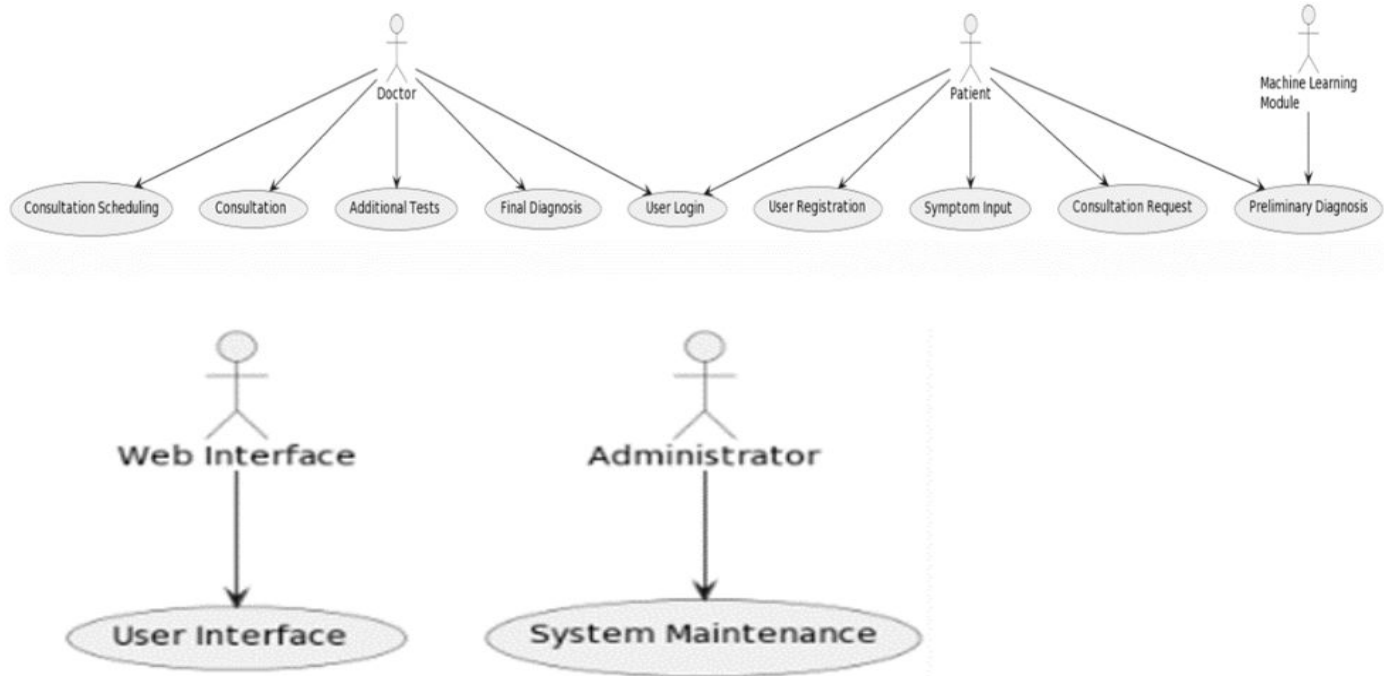
Sprint Report

We run a 2 week sprint cycle, which we document. Here is the sprint report for our latest sprint.

- Client Meeting Summary: We had a productive meeting with our client, Dr. Hajiarbabi, where we updated him on our current progress. The meeting focused on discussing the developments so far and the next steps in our project timeline.
- Model Evaluation Sprint: During this sprint, we evaluated four different models: Logistic Regression (LR), Support Vector Machine (SVM), Random Forest, and XGBoost. This evaluation was crucial in determining the most efficient and accurate model for our application.
- Model Selection and Integration: After thorough evaluation, XGBoost emerged as the chosen model due to its superior performance. We have initiated the process of integrating the XGBoost model with both the front-end and back-end components of our application.
- Front-End and Back-End Development: The front-end development is being done using HTML, CSS, and JavaScript, ensuring a user-friendly interface. For the back-end, we are utilizing Flask API to serve our XGBoost model, facilitating seamless interaction between the user interface and the model.

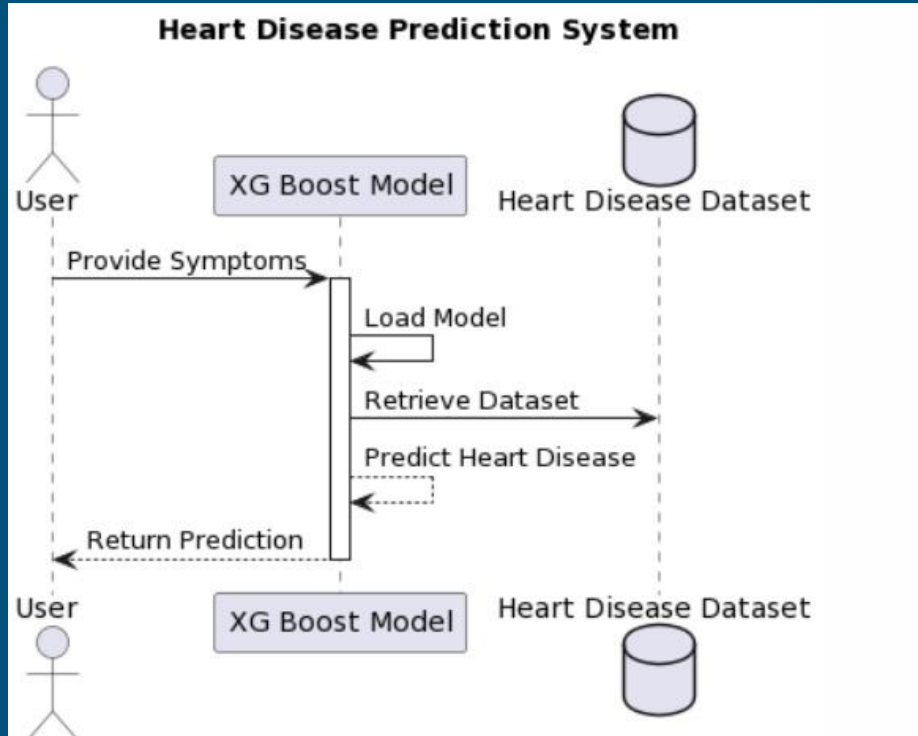
Use case Diagram

UCD:



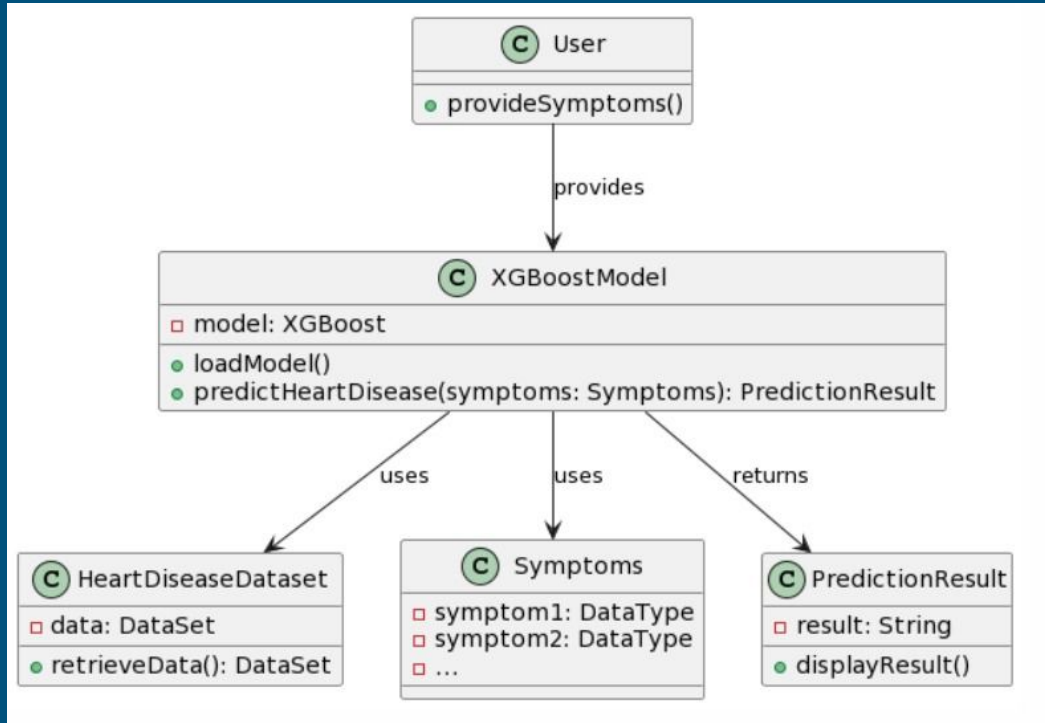
Sequence Diagram

SD:



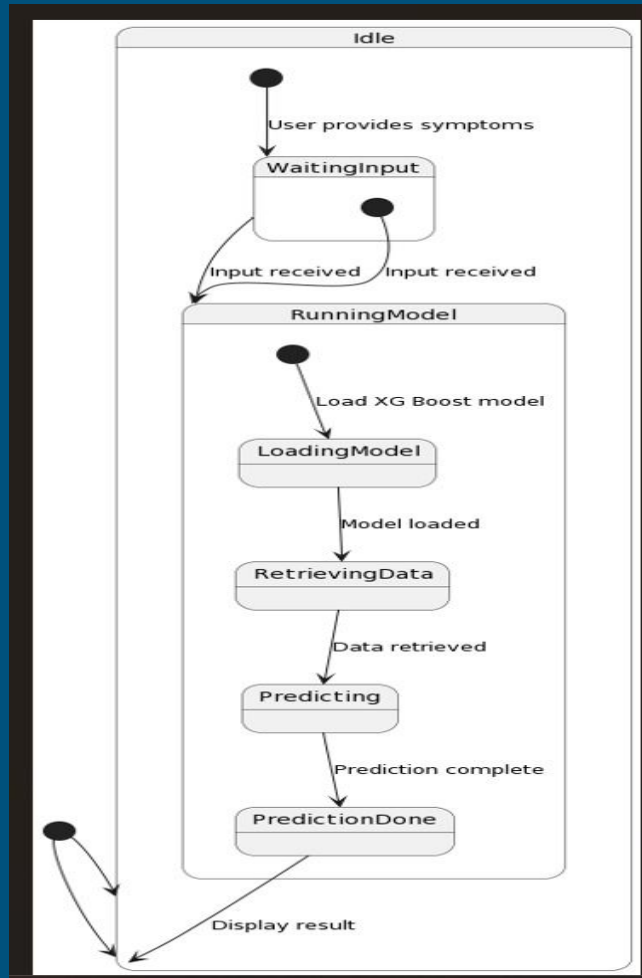
Class Diagram

CD:



State Diagram

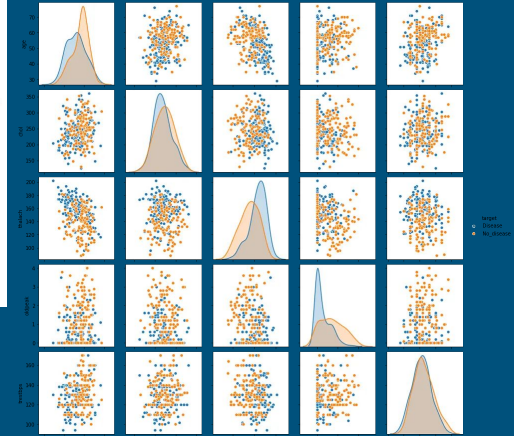
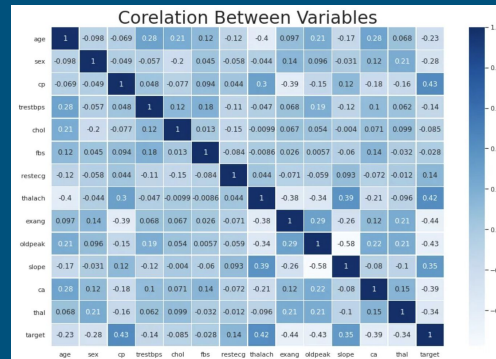
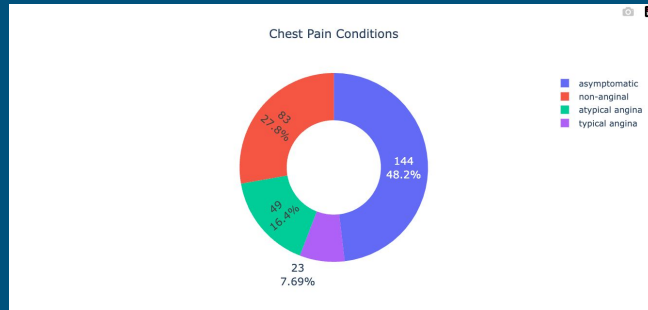
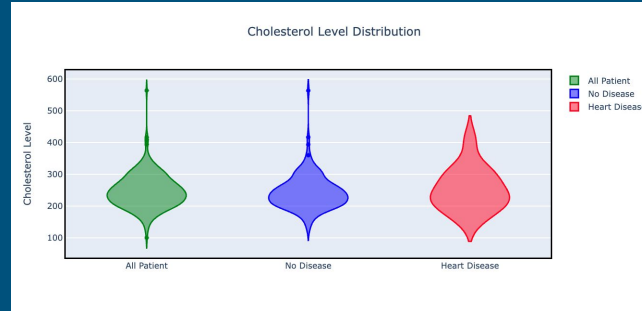
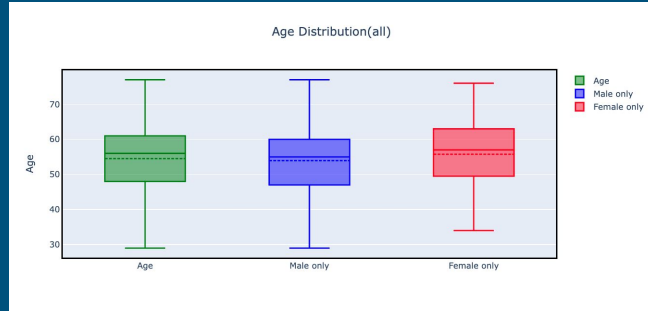
SD:



Model - Data

- Data is taken from actual ~1000 heart disease patients.
- Link to dataset - <https://archive.ics.uci.edu/dataset/45/heart+disease>
- All sensitive patient information like patient actual name, and social security numbers are removed from the dataset, and replaced with dummy values
- This database contains 76 attributes, but after careful PCA analysis as well as our use case, we decided to go with a subset of 14 features

Exploratory Data Analysis



Model Selection

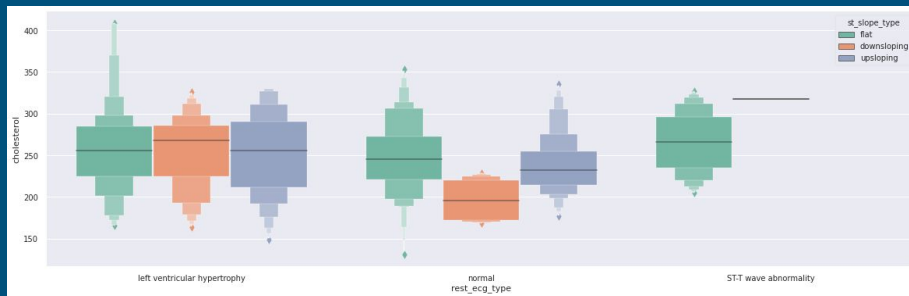
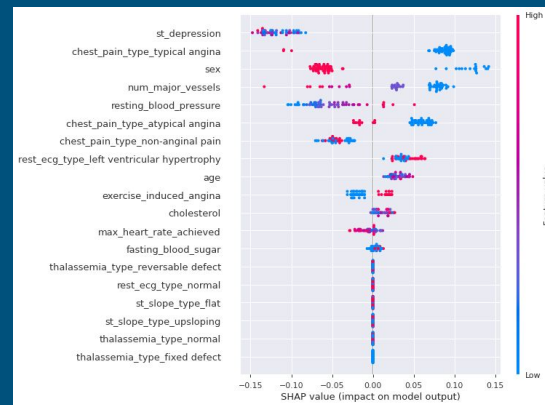
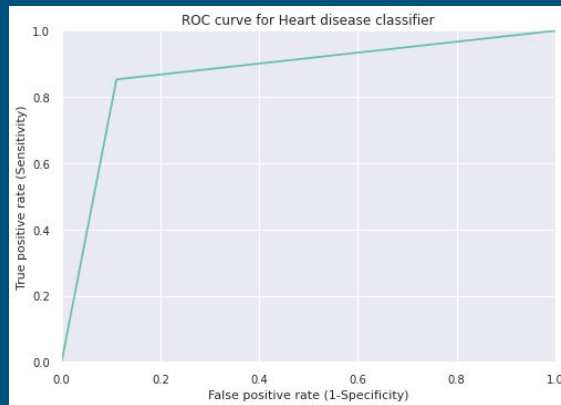
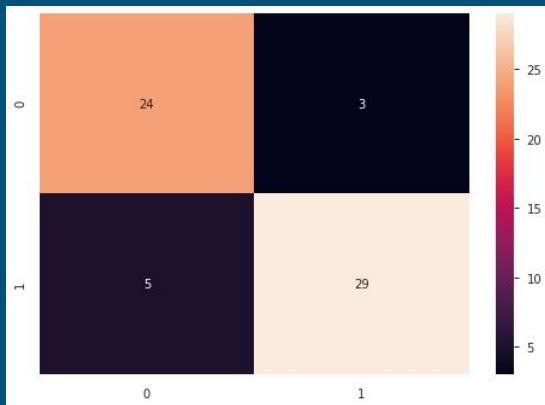
- We worked on a lot of data preprocessing, followed by feature ranking, feature selection and experimented with different models for our use case.



Model Name	Precision	Recall	F-1	Accuracy
Logistic Regression	0.83	0.85	0.9	0.9
SVM	0.7	0.76	0.73	0.78
Random Forest	0.88	0.85	0.87	0.85
XGBoost	0.92	0.95	0.93	0.92

Model Performance

We evaluated our models on various metrics -



Front-end

- The expert system is being built using HTML, CSS and Javascript.
- The work on the Prediction page is completed.
- The Login and Sign-up Authentication will be worked on and completed.

Front-end

Heart Disease Predictor

An Expert system Web Application that predicts chances of having a heart Disease

Age	<input type="text" value="Enter Your age"/>
Sex	<div>----select option----</div>
Chest Pain Type	<div>----select option----</div>
Resting Blood Pressure	<input type="text" value="A number in range [94-200] mmHg"/>
Serum Cholesterol	<input type="text" value="A number in range [126-564] mg/dl"/>
Fasting Blood Sugar	<div>----select option----</div>
Resting ECG Results	<div>----select option----</div>
Max Heart Rate	<input type="text" value="A number in range [71-202] bpm"/>
Exercise-induced Angina	<div>----select option----</div>
ST depression	<input type="text" value="ST depression, typically in [0-6.2]"/>
slope of the peak exercise ST segment	<div>----select option----</div>
Number of Major vessels	<input type="text" value="Typically in [0-4]"/>
Thalassemia	<div>----select option----</div>

Register & predict

Back-end and Integration

Flask API serving as the backend for our project, and when a user interacts with the front end by clicking on the predict button, the Flask API processes the request, runs a predictive model, and sends back the prediction percentage to be displayed on the front end. Here's a brief overview of the flow:

User Interaction:

- User clicks the predict button on the front-end interface.

Front-end Request:

- Front-end sends a request to the Flask API, indicating that a prediction is requested.

Flask API Processing:

- Flask API receives the request.
- It triggers the backend process to run the predictive model using the provided input or parameters.

Back-end and Integration

Model Prediction:

- The model processes the input and generates a prediction.

Flask API Response:

- Flask API sends the prediction percentage back to the front end as a response.

Front-end Display:

- Front-end receives the response from the Flask API.
- The prediction percentage is displayed on the front-end interface.

Tools and Development Environment

