

# **A PROJECT REPORT ON**

## **“Heart Disease Prediction System Using Machine Learning ”**

**BY**

Sakshi Shimpi 72214705J

Shreya Shingate 7214708C

Zeeshan Shaikh 72259406C

**UNDER THE GUIDANCE OF**

**Prof. Sarjare.M.D**



**DEPARTMENT OF AIML ENGI**

**Navsahyadri Education Society's Group of Institutions. Naigaon, Pune. 041**



**SAVITRIBAI PHULE PUNE UNIVERSITY**

**2022-23**



## **CERTIFICATE**

This is to certify that the project report entitled

**“Heart Disease Prediction System Using Machine Learning ”**

### **Submitted By**

Sakshi Shimpi      72214705J

Shreya Shingate      72214708C

Zeeshan Shaikh      72259406C

is a bonafide student of this institute and the work has been carried out by them under the supervision of **Prof. Sarjare.M.D** and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the award of the degree of Bachelor of Engineering (COMPUTER ENGINEERING).

**Prof.Sarjare.M.D**  
Internal Guide  
Dept. of AIML Engineering

**Prof. I. T. Mukherjee**  
Head Of Department  
Dept. AIML Engineering

**Prof. S. S. Shelar**  
Project Coordinator  
Place : Pune

**Dr.A.D.Lokhande**  
External Examiner

**Dr.A.D.Lokhande**  
Principal

Date :

## Acknowledgment

*It gives us great pleasure in presenting the project report on ‘**Heart Disease Prediction system Using Machine Learning**’.*

*We would like to take this opportunity to thank my internal guide **Prof. Sarjare.M.D** for giving me all the help and guidance we needed. we are really grateful to them for their kind support. Their valuable suggestions were very helpful.*

*We are also thankful to all staff members and our project coordinator **Prof. S.S.Shelar** for his valuable guidance.*

*We would also like to thank **Dr. A.D. Lokhande** for his for his unflinching help, support and cooperation during the project.*

*We are also grateful to **Prof.I.T.Mukherjee** , Head of AIML Engineering Department, **NGI, PUNE** for his indispensable support, suggestions.*

Sakshi Shimpi  
Shreya Shingate  
Zeeshan Shaikh

BE AIML Engg

## **Abstract**

We are living in the modern Technological era, and we are trying to solve the problem using technology for that we are striving for more knowledge with help of our technology, the more our lifestyle is changing. In the fast growing world everyone is neglecting their Health. Result of this Health diseases are increasing day by day due to our lifestyle and other hereditary reasons. Especially Heart Diseases have become a major issue these days. WHO report that over 11 million deaths are caused each year worldwide due to heart related complications. Using this we can understand that the condition is very serious. There is an urgent need for some technological solutions which can help in educating people and early diagnosis of these disease. This requires a string infrastructure including a large enough work force, which is a slow process. In the recent year machine learning has evolved a lot. To resolve this problem we can use Machine Learning that can help us in this. Sudden increase in the number of patients depict the need for scalability in our medical system. While work force may be limited, we can look for algorithmic solutions for screening and diagnosis stages, due to their correlation with Our project "Heart Health Classification and Early Diagnosis of Heart Disease" is a system that uses predictive capability of machine learning models based on similar data points collected in past. We are going to use the previous medical report data to predict the risk of heart disease. Using our system user can estimate the chance of being suffer from heart disease. Our system takes input data and runs 5 Machine Learning Models on the data and gives the result. We are trying to provide the user-friendly platform to user that can be access remotely to screening facilities that will predict the risk of heart disease and that will eliminate the load on the medical system.

# INDEX

Acknowledgment . . . . .	I
Abstract . . . . .	II
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Overview . . . . .	1
1.2 Motivation . . . . .	2
1.2.1 Objectives: . . . . .	3
1.3 Project Scope and Limitations . . . . .	3
1.4 Methodologies of Problem solving . . . . .	4
<b>2 LITERATURE SURVEY</b>	<b>5</b>
<b>3 SOFTWARE REQUIREMENTS SPECIFICATION</b>	<b>8</b>
3.1 Software Design . . . . .	8
3.1.1 User Interfaces . . . . .	8
3.1.2 Software Interfaces . . . . .	8
3.2 Non Functional Requirements . . . . .	9
3.2.1 Performance Requirements . . . . .	9
3.2.2 Safety Requirements . . . . .	9
3.2.3 Security Requirements . . . . .	9
3.2.4 Software Quality Attributes . . . . .	10
3.3 System Requirements . . . . .	12
3.3.1 Database Requirements . . . . .	12
3.3.2 Software Requirements . . . . .	12
3.4 Hardware Design . . . . .	12

3.4.1	Hardware Requirements .....	12
3.4.2	Hardware Interfaces .....	12
3.5	Analysis Models: SDLC Model to be applied.....	13
<b>4</b>	<b>SYSTEM DESIGN</b>	<b>14</b>
4.1	System Architecture .....	14
4.2	MATHEMATICAL MODEL .....	16
4.3	Data Flow Diagrams .....	17
4.3.1	Level 0 Data Flow Diagram.....	17
4.3.2	Level 1 Data Flow Diagram.....	18
4.4	UML Diagrams.....	19
4.4.1	Class Diagram.....	19
4.4.2	Use Case Diagram .....	20
4.4.3	Sequence Diagram.....	21
4.4.4	Activity Diagram .....	22
<b>5</b>	<b>PROJECT PLAN</b>	<b>23</b>
5.0.1	Reconciled Estimates.....	23
5.1	Project Estimate.....	23
5.1.1	COCOMO Model .....	23
5.1.2	Reconciled Estimates.....	27
5.1.3	Project Resources.....	28
5.2	Risk Management.....	29
5.3	Project Schedule .....	31
5.3.1	Project task set .....	31
5.4	Team Organization .....	33
5.4.1	Team Structure.....	33
<b>6</b>	<b>PROJECT IMPLEMENTATION</b>	<b>34</b>
6.1	Tools and Technologies Used: .....	34
6.2	Algorithm Details: .....	35
6.2.1	Random Forest.....	35
6.2.2	<b>Logistic Regression</b> .....	35

6.2.3	<b>K-Nearest Neighbors .....</b>	<b>35</b>
6.2.4	<b>Decision Tree.....</b>	<b>36</b>
<b>7</b>	<b>SOFTWARE TESTING .....</b>	<b>37</b>
7.1	Type of Testing.....	37
7.2	Test Cases and Test Results .....	38
<b>8</b>	<b>RESULTS .....</b>	<b>40</b>
8.1	Outcomes.....	40
8.2	Screenshot .....	40
<b>9</b>	<b>CONCLUSION .....</b>	<b>43</b>
9.1	Conclusion.....	43
9.2	Future Scope.....	43
9.3	Applications.....	44
<b>10</b>	<b>REFERENCES .....</b>	<b>45</b>
 <b>Annexure A Appendix: Publications</b>		
 <b>Annexure B Appendix: Certificates</b>		
 <b>Annexure C Appendix: Source Code</b>		
 <b>Annexure D Appendix: Plagiarism Report</b>		

# List of Figures

1.1	Overview . . . . .	1
1.2	Overview . . . . .	2
3.1	Waterfall Model.....	13
4.1	System Architecture .....	14
4.2	Level 0 Data Flow Diagram.....	17
4.3	Level 1 Data Flow Diagram.....	18
4.4	Class Diagram .....	19
4.5	Use Case Diagram .....	20
4.6	Sequence Diagram.....	21
4.7	Activity Diagram .....	22



# List of Tables

5.1	Project Estimate.....	24
5.2	Time line Chart.....	33

# CHAPTER 1

## INTRODUCTION

### 1.1 OVERVIEW

Increasing population, changing life patterns, new diseases, pandemics, and change in the natural environment has surely changed the way we take care of our health. Maintaining good health is a major factor associated with any individual's efficiency and It has always been a big concern for all the countries of the world. Providing a good medical service across the whole country proves to be challenging for developing and poor countries.

Medical doctors (per 10,000)

WHO

Last updated: 2022-01-24

EXPORT DATA in CSV format

Request here & Save file

Location	Medical doctors (per 10,000)	Medical doctors (number)	Generalist medical practitioners (number)	Specialist medical practitioners (number)	Medical facilities (number)
World	10.00	10,000,000	10,000,000	10,000,000	10,000,000
High income	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Upper middle income	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Lower middle income	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Least developed countries: UN classification	10.00	10,000,000	10,000,000	10,000,000	10,000,000
European Region	10.00	10,000,000	10,000,000	10,000,000	10,000,000
South-East Asia Region	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Africa Region	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Americas Region	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Eastern Mediterranean Region	10.00	10,000,000	10,000,000	10,000,000	10,000,000
World	10.00	10,000,000	10,000,000	10,000,000	10,000,000
United States of America	24.36	10,000,000	10,000,000	10,000,000	10,000,000
France	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Germany	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Japan	10.00	10,000,000	10,000,000	10,000,000	10,000,000
South Korea	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Sweden	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Denmark	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Netherlands	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Switzerland	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Australia	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Canada	10.00	10,000,000	10,000,000	10,000,000	10,000,000
United Kingdom	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Italy	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Spain	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Belgium	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Austria	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Portugal	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Greece	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Ireland	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Finland	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Poland	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Czechia	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Slovakia	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Hungary	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Slovenia	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Lithuania	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Latvia	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Estonia	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Malta	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Cyprus	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Israel	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Lebanon	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Yemen	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Saudi Arabia	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Qatar	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Ukraine	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Romania	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Bulgaria	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Serbia	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Croatia	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Bosnia and Herzegovina	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Montenegro	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Albania	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Moldova	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Belarus	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Georgia	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Armenia	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Azerbaijan	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Kazakhstan	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Kyrgyzstan	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Tajikistan	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Uzbekistan	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Turkmenistan	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Trinidad and Tobago	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Barbados	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Antigua and Barbuda	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Nevis	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Vincent and the Grenadines	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Grenada	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Kitts and Nevis	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Lucia	10.00	10,000,000	10,000,000	10,000,000	10,000,000
Dominica	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Eustace	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. John	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. James	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Mary	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Peter	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Paul	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Andrew	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. David	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Elizabeth	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. James	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. John	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Mary	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Peter	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Paul	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Andrew	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. David	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Elizabeth	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. James	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. John	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Mary	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Peter	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Paul	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Andrew	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. David	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Elizabeth	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. James	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. John	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Mary	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Peter	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Paul	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Andrew	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. David	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Elizabeth	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. James	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. John	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Mary	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Peter	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Paul	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Andrew	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. David	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Elizabeth	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. James	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. John	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Mary	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Peter	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Paul	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Andrew	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. David	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Elizabeth	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. James	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. John	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Mary	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Peter	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Paul	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Andrew	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. David	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Elizabeth	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. George	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. James	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. John	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Mary	10.00	10,000,000	10,000,000	10,000,000	10,000,000
St. Peter	10.00	10,000,000	10,000,000	10	

(the USA and Western European) have this ratio to a good level ( 25:10000), for Southeast Asian countries like India are way below the world average (India has a ratio of 12.21 to the 10000, whereas the world average is closer to 10:10000) [1]

Medical doctors (per 10,000)

World Bank

EXPORT DATA in CSV format  
[Request here](#) & [Save file](#)

Last updated: 2022-09-28

Location	Medical doctors (per 10,000)	Medical doctors (number)	General medical practitioners (number)	Specialist medical practitioners (number)	Medical assistants (number)
Armenia	2.24	1,000	1,000	0	0
Australia	6.17	2,752	1,465	1,287	0
Austria	5.46	2,444	1,819	625	0
Bahrain	9.00	4,000	200	3,800	0
Bangladesh and Nepal	0.25	1,000	1,000	0	0
Barbados	16.24	7,200	4,400	2,800	0
Belarus	10.00	45,000	30,000	15,000	0
Belize	22.73	10,000	4,000	6,000	0
Bhutan	12.00	60	60	0	0
Bolivia	5.00	21,000	12,000	9,000	0
Bosnia	20.00	100,000	50,000	50,000	0
Brazil	73.0	15,000	15,000	0	0
British Virgin Islands	16.0	1,000	1,000	0	0
United Kingdom of Great Britain and Northern Ireland	16.0	60,000	30,000	30,000	0
United Republic of Rwanda	1.24	1,000	1,000	0	0
<b>United States of America</b>	<b>24.26</b>	<b>105,079</b>	<b>52,071</b>	<b>53,008</b>	<b>0</b>
Uruguay	11.11	5,000	2,000	3,000	0
Uzbekistan	24.74	10,000	10,000	0	0
Venezuela	7.14	300	300	0	0
Yemen (Democratic Republic of)	70.0	40,000	40,000	0	0
Yemen	3.00	10,000	10,000	0	0
Zambia	7.00	3,000	1,500	1,500	0
Zimbabwe	2.00	1,000	1,000	0	0

Figure 1.2: Overview

## 1.2 MOTIVATION

Increasing this ratio of doctors to patients is a complex process and takes a lot of time in order of years or maybe decades and medical buildings equipped with modern machinery are needed. It is also a general foreseen that people avoid going to doctors for preliminary stages of their diseases and other risks. Designing a suitable system for them so that they can pre-screen themselves can be very helpful hence A basic solution is to design a good scalable system here. Scalability here means that the technological stacks used can expand the scope of health care without needing more workforce i.e. Use software services. Software services generally are beneficial for the fact that they can detect Early Disease and their Speed Precision in Medical Diagnostics, enabling the Provision of Quality Care. It also promotes Improved Patient Participation Management and improves the chances of safer Treatment Solutions. A good scalable software service ensures that in case of larger participation need, it can take release loads from its concerned department. Since we know that building smarter and better infrastructure with better equipment, training more workforce, etc. are time taking processes. Once made available, they cannot be rolled back and

are limited but they are permanent. They obviously increase the capacity of the medical system but don't contribute to increasing scalability at the current level which is the need of time. Hence, we are looking at their non- biological counterparts, i.e., computers, algorithms, to help us. If we consider any diagnosis of a patient, then it is greatly affected by experience, knowledge of its doctor. Some decisions are so intense that it may take suggestions from a few other trained and experienced people of the concerned area to arrive at a decision, it is the stages that need huge amounts of experience with little analysis. It can be solved easily by making use of computer based on previous medical record, etc. The reason for this is that scaling-up only needs more computing power which is much cheaper and available than training humans for the same, and can be deployed much more quickly. Before all, a screening phase should exist which can be predicted just by asking simple questions, simple symptoms, if they exist etc.

### **1.2.1 Objectives:**

1. To learn, understand and use Python programming language.
2. To learn, understand and use machine learning algorithms for prediction.
3. To collect and preprocess medical data related to heart disease.
4. To develop a user-friendly GUI for heart disease prediction.
5. To implement a system that predicts the likelihood of heart disease based on input symptoms and health data.
6. To evaluate the system's accuracy and improve model performance using real-world datasets.

## **1.3 PROJECT SCOPE AND LIMITATIONS**

Project will be developed as a prototype model using python programming language. It will run as a windows application.

## **1.4 METHODOLOGIES OF PROBLEM SOLVING**

Deep Learning: Deep structured learning or hierarchical learning or deep learning in short is part of the family of machine learning methods which are themselves a subset of the broader field of Artificial Intelligence. Deep learning is a class of machine learning algorithms that use several layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input. Deep neural networks, deep belief networks and recurrent neural networks have been applied to fields such as computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, and bioinformatics where they produced results comparable to and in some cases better than human experts have.

Deep Learning Algorithms and Networks are based on the unsupervised learning of multiple levels of features or representations of the data. Higher-level features are derived from lower level features to form a hierarchical representation. use some form of gradient descent for training.

# **CHAPTER 2**

## **LITERATURE SURVEY**

Heart disease is one of the most critical and life-threatening conditions that affect millions of people worldwide. It remains a leading cause of death in both developed and developing countries. With the growth of Artificial Intelligence (AI) and Machine Learning (ML), researchers have turned to intelligent systems to help in the early detection and prediction of heart-related illnesses. A large number of studies have been carried out using data mining and machine learning algorithms to enhance the decision-making capabilities of healthcare professionals. This literature survey aims to present a comprehensive review of the previous work done by researchers in the area of heart disease prediction using various machine learning and AI techniques.

In 2008, Palaniappan and Awang Nor developed an Intelligent Heart Disease Prediction System using data mining techniques. They applied algorithms such as Naive Bayes, Decision Trees, and Neural Networks\* to patient datasets. Their study aimed to support medical experts by analyzing historical patient data to identify disease patterns. Among the models used, the Naive Bayes algorithm showed the highest prediction accuracy, suggesting its suitability for classification-based medical applications.

Soni et al. (2011) further enhanced the work by implementing Decision Tree and Naive Bayes classifiers using the Cleveland Heart Disease dataset, which is one of the most popular data mining software. The findings concluded that Decision Trees gave better results in terms of classification accuracy, recall, and precision, which are crucial for medical predictions. Their research helped validate the useful-

ness of basic machine learning models in clinical data analysis.

In 2014, Chaurasia and Pal conducted a comparative study on different classification techniques such as K-Nearest Neighbor (KNN), Naive Bayes, and Decision Tree algorithms for heart disease prediction. They discovered that Decision Tree classifiers achieved the highest accuracy on their dataset. Additionally, the study emphasized the importance of feature selection, which means choosing the most relevant input attributes (like blood pressure, cholesterol levels, etc.) that directly impact the model's performance. Their work highlighted that irrelevant or noisy features could reduce accuracy, and proper data preprocessing is essential.

Another significant contribution was made by Kumar and Reddy (2015), who conducted a broad survey of heart disease prediction models. They reviewed multiple research papers and outlined common problems faced by researchers, such as missing data, imbalanced datasets, and overfitting of models. They recommended using ensemble techniques, especially Random Forest and Gradient Boosting, as these models combine multiple weak learners to build a stronger and more accurate prediction system. Their survey played an important role in showing the real-world challenges of working with healthcare data and building reliable AI models. As technology continued to evolve, researchers began using modern deep learning approaches. Deep Neural Networks (DNN), Recurrent Neural Networks (RNN),

and Convolutional Neural Networks (CNN) have recently been applied in medical data analysis, including heart disease detection. These deep models can learn complex patterns and learning relationships from data that simpler algorithms might miss. However, deep learning requires large volumes of high-quality data and high-performance computing resources, making them less suitable in environments where computational power or medical data is limited. Despite this, some recent studies have achieved prediction accuracies of over 90

In 2019, Khan et al. introduced a hybrid model combining Support Vector Machines (SVM) with optimization algorithms like Genetic Algorithms (GA) to fine-tune parameters and improve model accuracy. They found that hybrid models not only improved accuracy but also reduced false positives, which is a major concern in medical predictions. False predictions in heart disease can have serious

consequences, so improving the precision of AI models is extremely important.

In 2020, Mohammed et al. applied Logistic Regression, SVM, and Random Forest classifiers on multiple datasets including Cleveland, Statlog, and Hungarian datasets. They compared results across models and found that Random Forest consistently performed well across all datasets. This suggests that Random Forest is a reliable and robust algorithm for heart disease prediction in real-world scenarios.

Additionally, feature engineering and dimensionality reduction techniques like Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE) have been used in many studies to reduce complexity and increase the interpretability of the models.

These techniques help in simplifying the input data and removing less useful features, thereby improving the model's performance and reducing training time.

Researchers have also explored the use of Internet of Things (IoT) devices and wearable technology to collect real-time patient data like heart rate, ECG, and blood pressure. These data streams, when combined with machine learning models, can help build real-time heart disease monitoring systems. For instance, combining ML with smartwatches or fitness trackers can allow for continuous monitoring and early warning in high-risk patients.

In summary, the literature reviewed shows that various AI and ML techniques have been applied for heart disease prediction with promising results. Models such as Decision Trees, Naive Bayes, Logistic Regression, Support Vector Machines, KNN, and Random Forest are commonly used due to their ease of implementation and good accuracy. Deep learning models are also gaining popularity, although they require more resources. Data preprocessing, feature selection, and model tuning play a vital role in building efficient prediction systems. While AI models are not yet a complete replacement for medical experts, they can act as powerful tools to assist in early diagnosis, reduce human error, and improve patient outcomes.



# **CHAPTER 3**

## **SOFTWARE REQUIREMENTS SPECIFICATION**

### **3.1 SOFTWARE DESIGN**

#### **3.1.1 User Interfaces**

The user interface of the Heart Disease Prediction System is designed to be simple, clean, and userfriendly to allow easy interaction for both technical and non-technical users. A web-based interface is created using Streamlit, where users can enter patient medical details such as age, gender, chest pain type, blood pressure, cholesterol, fasting blood sugar, ECG results, heart rate, exercise-induced angina, ST depression, number of major vessels, and thalassemia type. All inputs are taken through text fields, dropdowns, checkboxes, and radio buttons. Once the user clicks the "Predict" button, the system processes the inputs and displays whether the person is at risk of heart disease or not. The prediction result is shown along with an optional confidence score. The interface ensures smooth usability and supports accurate medical decision-making

#### **3.1.2 Software Interfaces**

The Heart Disease Prediction system is developed using Python and is integrated with a simple and interactive interface built using Streamlit. The software interface allows users to input medical parameters and receive a prediction result in real-time. The frontend is web-based, running locally or on a cloud server, and interacts

with the backend machine learning model. The backend processes the input data, performs prediction using a trained model (e.g., Logistic Regression, Random Forest, etc.), and returns the result to the frontend. Libraries like pandas, scikit-learn, and joblib are used for data processing and model deployment. The interface is lightweight, responsive, and works across all modern browsers, providing a smooth and effective user experience.

## **3.2 NON FUNCTIONAL REQUIREMENTS**

### **3.2.1 Performance Requirements**

- The Heart Disease Prediction System must provide fast and reliable predictions to ensure a smooth user experience. The system is expected to generate results within 2–3 seconds after the user submits the input data. It should handle multiple prediction requests efficiently without crashing or slowing down. The machine learning model should maintain a prediction accuracy of at least 85

### **3.2.2 Safety Requirements**

- The Heart Disease Prediction System handles sensitive health-related information, so it must ensure data safety and prevent any harm caused by incorrect predictions. Although the system is designed to assist in medical diagnosis, it should clearly display a disclaimer that it is not a substitute for professional medical advice. The system must be protected against unauthorized access, and no patient data should be permanently stored without consent. In case of system errors or crashes, proper error messages should be shown, and the application should safely terminate without data loss or misleading outputs. This ensures both user safety and system reliability.

### **3.2.3 Security Requirements**

- • The system must protect patient data by implementing secure access controls and encryption during data transmission. User authentication should prevent

unauthorized access. Additionally, sensitive information must not be stored without explicit user consent to ensure privacy compliance.

### 3.2.4 Software Quality Attributes

- **Usability:**

Usability refers to how easy and intuitive the Heart Disease Prediction system is for users, especially doctors and patients who may not have technical expertise. The interface is designed to be simple, with clear input fields and easy-to-understand output results. Instructions and error messages are provided to guide users smoothly through the prediction process. High usability ensures users can efficiently operate the system without confusion, making it accessible to a wider audience and improving overall satisfaction.

- **Reliability:**

Reliability ensures that the Heart Disease Prediction system consistently performs its functions accurately without failure over time. The system should provide stable and dependable predictions even under varying input conditions. It must handle errors gracefully and recover quickly from unexpected issues to avoid incorrect results. High reliability builds user trust, especially in critical healthcare applications where decision-making depends on accurate predictions.

- **Performance:**

Performance refers to how quickly and efficiently the Heart Disease Prediction system processes input data and delivers results. The system should provide predictions within a few seconds to ensure a smooth user experience. It must efficiently use computational resources to handle multiple requests without delays or crashes. Good performance is essential to make the system practical for real-time healthcare applications.

- **Security :**

Security ensures that the Heart Disease Prediction system protects sensitive patient data from unauthorized access and breaches. It involves implement-

ing user authentication, data encryption, and secure communication protocols. Maintaining data privacy and confidentiality is critical to comply with health-care regulations and build user trust.

### **3.3 SYSTEM REQUIREMENTS**

#### **3.3.1 Database Requirements**

##### **MySQL Database**

MySQL is an open source database which is mainly a RDBMS i.e. relational database management system. As a database server, primary function of this software is to store and retrieve data as requested by other software applications like Java which may or may not run either on the same computer or on a different computer. This can be across the network either in internet or intranet.

#### **3.3.2 Software Requirements**

1. **Operating System:** Microsoft Windows 7 and Above
2. **Programming Language:** Python
3. **IDE:** Python IDLE
4. **Libraries :** pandas, numpy, scikit-learn, matplotlib, joblib, and Streamlit or Flask for UI.

### **3.4 HARDWARE DESIGN**

#### **3.4.1 Hardware Requirements**

1. **Processor:** Intel Core I3 or Higher
2. **RAM:** 4 GB or Higher
3. **Hard Disk:** 100 GB (min)

#### **3.4.2 Hardware Interfaces**

A hardware interface is needed to run the software. Python, Keras, Numpy compatible hardware is required which is minimal requirement.

### 3.5 ANALYSIS MODELS: SDLC MODEL TO BE APPLIED

#### SDLC model to be applied

##### Waterfall Model:

The Waterfall Model is among very first and old model of software development life cycle. It is also called as a linear-sequential life cycle model. This is very simple in nature and easy to understand or use. This is step by step method so next step can only be begin once earlier has been completed. This is mainly used for small scale project. Constant or fixed requirement should be there for this type of model.

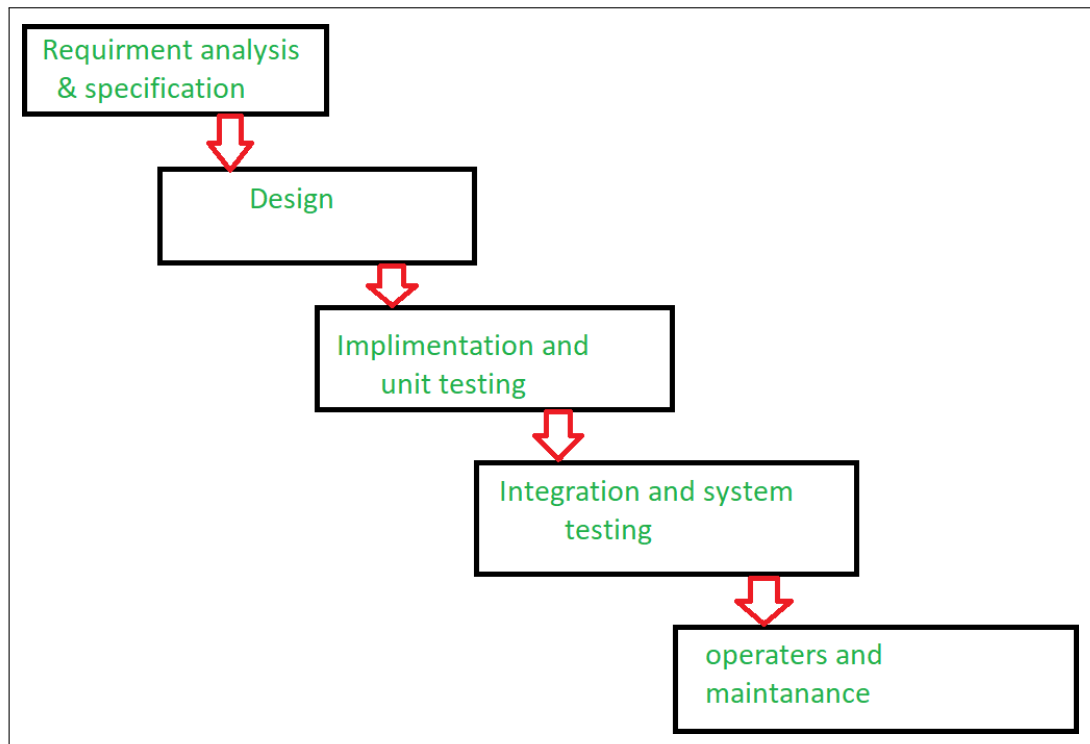


Figure 3.1: Waterfall Model

# CHAPTER 4

## SYSTEM DESIGN

### 4.1 SYSTEM ARCHITECTURE

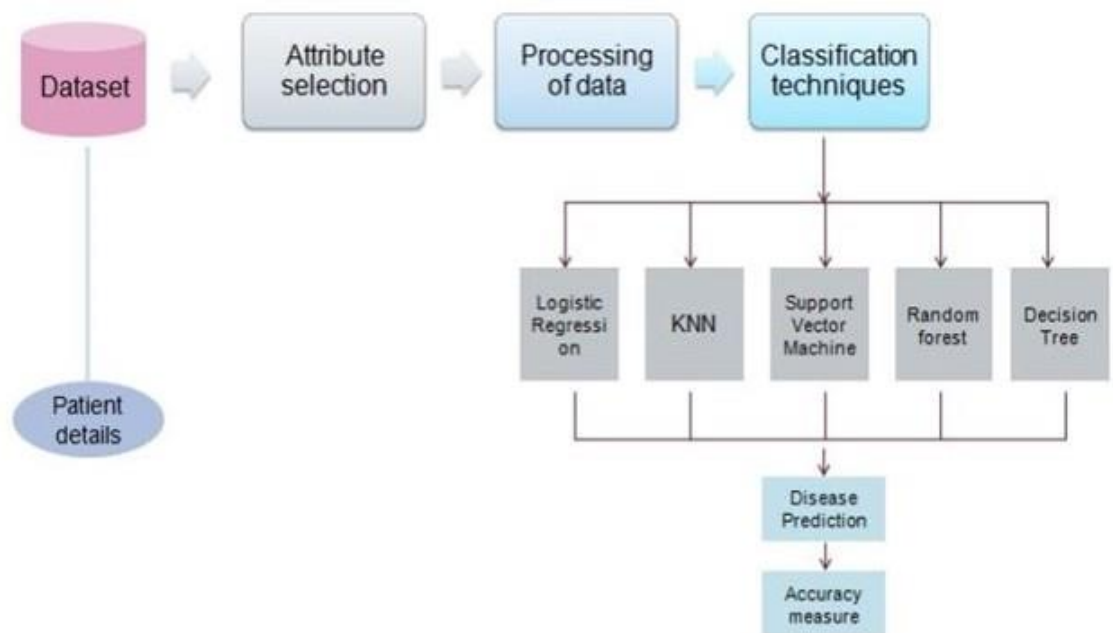


Figure 4.1: System Architecture

The system architecture of the Heart Disease Prediction project is designed to be modular, scalable, and efficient to ensure smooth performance and accurate predictions. It primarily follows a three-tier structure: Data Layer, Model Layer, and Presentation Layer, each with a specific role in the system's operation. The Data Layer forms the foundation of the system and is responsible for

collecting and managing the dataset used for training and testing the prediction model. In this project, the UCI Cleveland Heart Disease Dataset is used, which contains essential patient attributes such as age, sex, chest pain type, cholesterol level, blood pressure, and more. The raw dataset is passed through several preprocessing steps such as handling missing values, converting categorical data to numerical format, normalizing values, and selecting important features that contribute most to prediction accuracy. This ensures that the input data is clean and ready for model training. The Model Layer is the core of the system where machine learning algorithms are applied. Algorithms like Logistic Regression, Random Forest, Support Vector Machine (SVM), or K-Nearest Neighbors (KNN) can be trained using the preprocessed data. The chosen model is evaluated using metrics such as accuracy, precision, recall, and F1-score to ensure high performance. Once the best-performing model is identified, it is saved using a serialization tool like joblib for later use in real-time predictions. This layer also handles the logic to load the model and generate predictions based on user input. The Presentation Layer serves as the front-end interface for the users, built using Streamlit or Flask, allowing doctors, patients, or researchers to interact with the system. It consists of input fields for entering patient details and a "Predict" button that triggers the model to analyze the data and return a prediction. The result is shown to the user as either "High Risk" or "Low Risk" of heart disease, often along with a confidence score. The interface is designed to be intuitive and user-friendly, with proper labels, error handling, and fast response time to ensure a good user experience. Overall, this system architecture ensures that the heart disease prediction system is easy to use, reliable in its predictions, and flexible for further development. Each layer is independent yet connected, making the system maintainable and extendable for future medical applications like diabetes or cancer prediction.



## 4.2 MATHEMATICAL MODEL

Let

1. Let  $I$  be the set of inputs:  $I = \text{age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal}$
2. Let  $O$  be the output:  $O = 0, 1$ , where  $0 = \text{no heart disease}$ ,  $1 = \text{presence of heart disease}$
3. Function  $f(I) \rightarrow O$  This means a machine learning function takes input  $I$  and predicts output  $O$  using trained models like Logistic Regression, Random Forest, etc.

Objects:

1. Input Objects ( $I$ ):

These are the parameters collected from the patient:

• Age– Patient’s age • Sex – Gender ( $0 = \text{Female}$ ,  $1 = \text{Male}$ ) • cp– Chest pain type ( $0$  to  $3$ ) • trestbps – Resting blood pressure • chol – Serum cholesterol (mg/dl) • fbs – Fasting blood sugar  $\geq 120$  mg/dl ( $1 = \text{True}$ ,  $0 = \text{False}$ ) • restecg – Resting ECG results ( $0, 1, 2$ ) • thalach – Maximum heart rate achieved • exang – Exercise-induced angina ( $1 = \text{Yes}$ ,  $0 = \text{No}$ ) • oldpeak – ST depression induced by exercise • slope – Slope of the ST segment ( $0, 1, 2$ ) • ca – Number of major vessels ( $0$  to  $3$ ) • thal – Thalassemia ( $0 = \text{Normal}$ ,  $1 = \text{Fixed defect}$ ,  $2 = \text{Reversible defect}$ )

Set of inputs:  $I = \text{age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal}$

. Output Object ( $O$ ):

$O = 0, 1$ , where  $0 \rightarrow \text{No heart disease}$   $1 \rightarrow \text{Presence of heart disease}$

. Function:

The system uses a function  $f(I) \rightarrow O$  This means: Based on input values  $I$ , the machine learning model predicts the result  $O$

### 4.3 DATA FLOW DIAGRAMS

A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing.

#### 4.3.1 Level 0 Data Flow Diagram

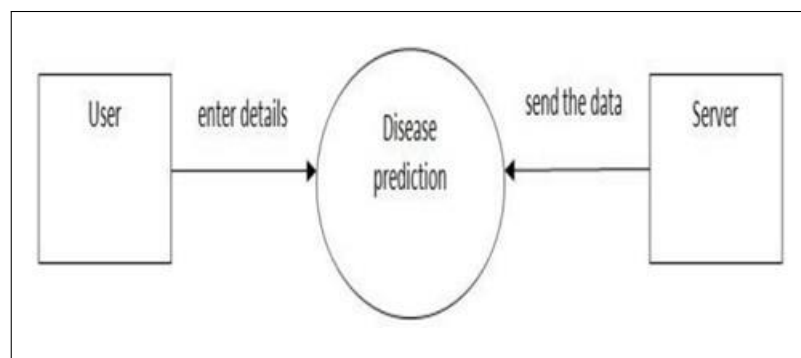


Figure 4.2: Level 0 Data Flow Diagram

### 4.3.2 Level 1 Data Flow Diagram



Figure 4.3: Level 1 Data Flow Diagram

## 4.4 UML DIAGRAMS

### 4.4.1 Class Diagram

A class diagram in the world of Unified Modeling Language or UML can be defined as a type of static structure diagram which mainly defines the structure of a system. It works by showing the system’s classes and their attributes and operations or methods also the relationships among objects.

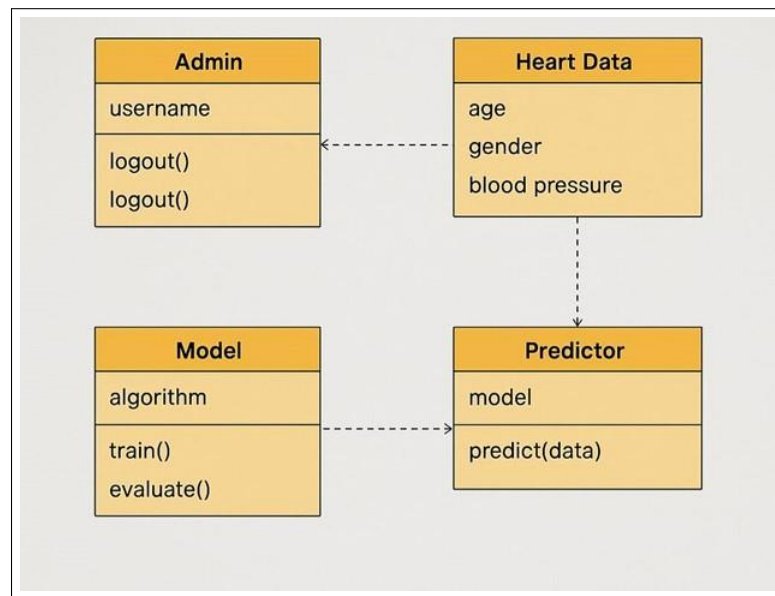


Figure 4.4: Class Diagram

#### 4.4.2 Use Case Diagram

Dynamic behavior is most important aspect to capture the model of any system. Dynamic behavior can be defined as the behavior of the system when it is running or operating. Static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior.

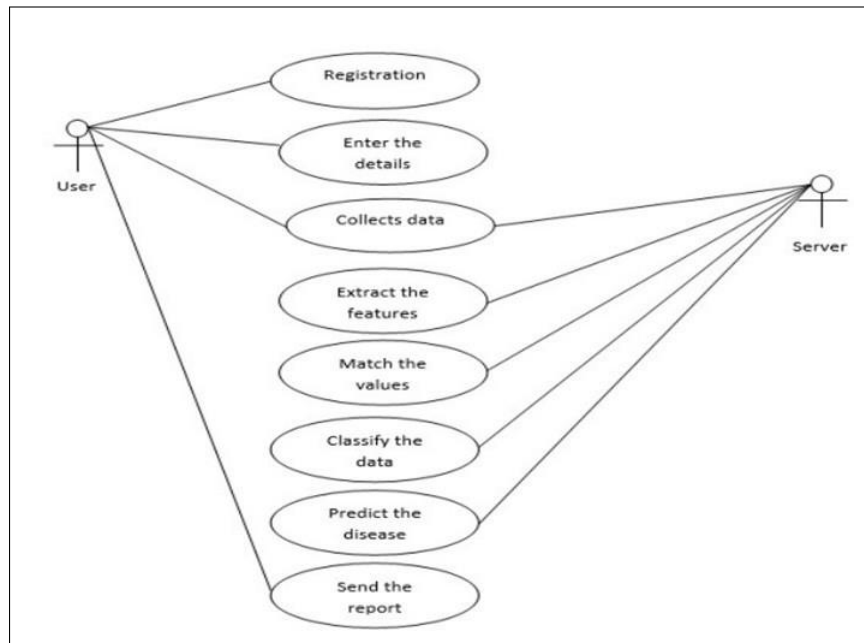


Figure 4.5: Use Case Diagram

### 4.4.3 Sequence Diagram

Sequence diagrams can be used to provide a graphical representation of object interactions or object coordination over the time. These basically displays a actor or user, and the objects and components they interact with in the execution of a use case. The sequence diagrams displays the own of messages from one object to another object, and as such correspond to the methods and events supported by a class/object.

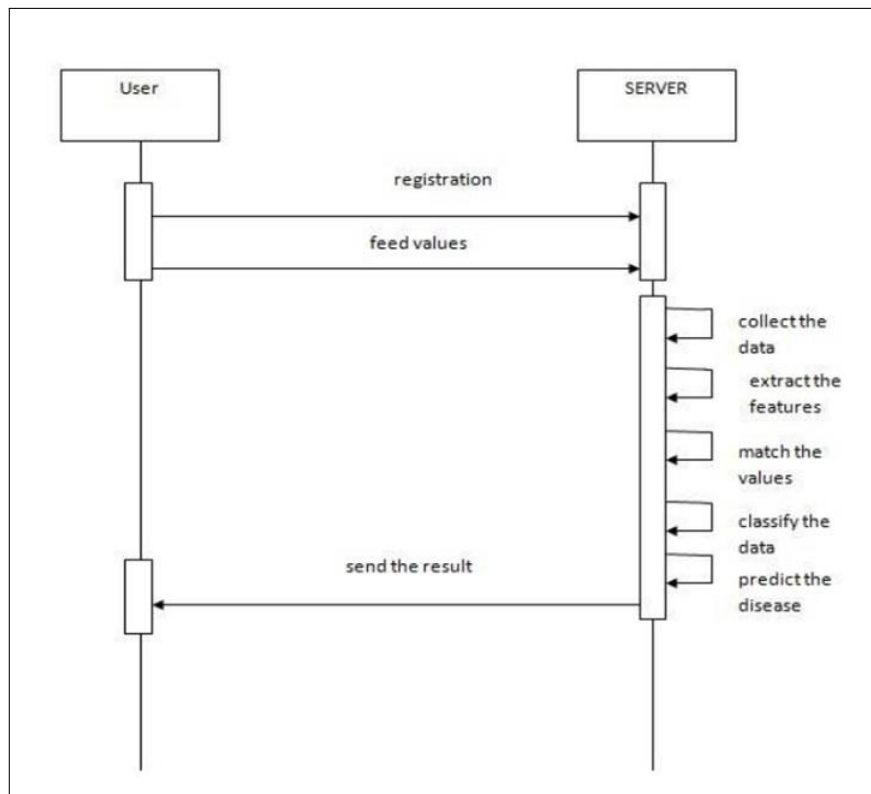


Figure 4.6: Sequence Diagram

#### 4.4.4 Activity Diagram

Activity diagram can be defined as a flowchart to display the flow from one activity to another activity. These activities could be described as an operation of the system. The control flow usually is drawn from one operation of application to another. This can be branched or sequential, or concurrent also. Activity diagrams can deal with all or many type of flow control and used different elements such as join or fork.

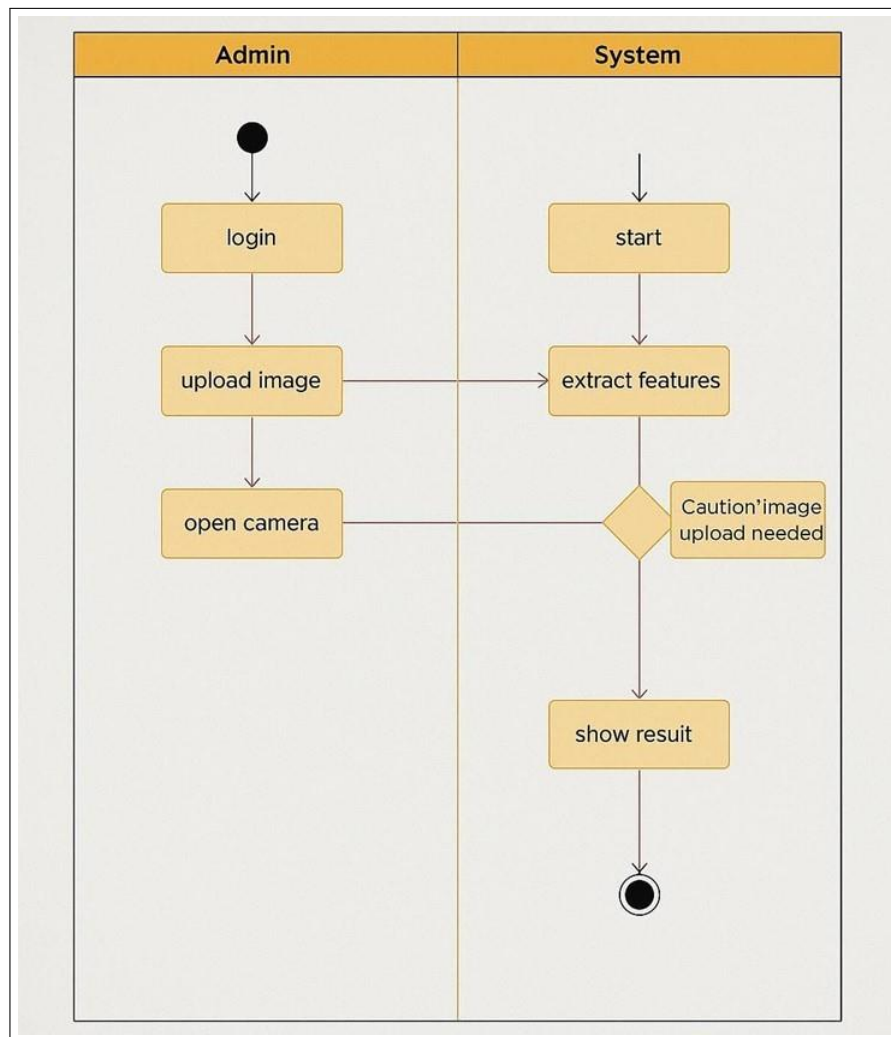


Figure 4.7: Activity Diagram

# CHAPTER 5

## PROJECT PLAN

Phase	Task	Description
Phase 1	Analysis	Study heart disease data and gather system requirements.
Phase 2	System Design	Design system flow, modules, and user interface.
Phase 3	Implementation	Develop and integrate all modules using machine learning.
Phase 4	Testing	Test system accuracy and fix any issues.
Phase 5	Maintenance	Improve and update the system after deployment.

### 5.0.1 Reconciled Estimates

## 5.1 PROJECT ESTIMATE

### 5.1.1 COCOMO Model

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e number of Lines of Code. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality. It was proposed by Barry Boehm in 1970 and is based on the



Sr.No.	Milestone Name	Milestone Description
1.	Requirement Analysis	Complete specification of system
2.	High level design	Identify the modules and the different entities and their relationships
3.	Detailed design	GUI design, program specification etc
4.	Build	Writing code for different modules
5.	Testing	Test the different modules together
6.	Final Review and Deployment	Checking all the requirements are fulfilled

Table 5.1: Project Estimate

study of 63 projects, which make it one of the best-documented models. The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort & Schedule:

- Effort: Amount of labor that will be required to complete a task. It is measured in person-months units.
- Schedule: Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.

Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below.

Boehm’s definition of organic, semidetached, and embedded systems:

1. Organic – A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.
2. Semi-detached – A software project is said to be a Semi-detached type if the vital characteristics such as team-size, experience, knowledge of the various programming environment lie in between that of organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered of Semi-Detached type.
3. Embedded – A software project with requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

All the above system types utilize different values of the constants used in Effort Calculations. Types of Models: COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. Any of the three forms can be adopted according to our requirements. These are types of COCOMO model:

1. Basic COCOMO Model The first level, Basic COCOMO can be used for quick and slightly rough calculations of Software Costs. Its accuracy is somewhat restricted due to the absence of sufficient factor considerations.

The basic COCOMO model estimates the software development effort using only lines of code.

$$E = a(KLOC)^b$$

Where,

E is the efforts applied by person in months,  $a = 3.0$  and  $b = 1.12$ , then

$$KLOC = 2.25$$

$$\text{Hence Efforts} = 3.0 (1.8)^{1.12},$$

$$E = 5.79 \text{ Person-month}$$

$$E = 6 \text{ Person-month}$$

Total of 6 Person-Month are required to complete the project successfully.

$$D = cb(E)^{db}$$

Where,

D = Development time in chronological months,  $cb = 2.5$  and  $db = 0.35$ ,

and  $E = 6 \text{ Person-Month}$

$$\text{Hence, Development Time} = 2.5 (1.8)^{0.35}$$

$$D = 3.07 \text{ months}$$

The approximate duration of project is 3 months.

$$P = E/D$$

Where,

P = Number of persons to accomplish project.

Hence, Number of Persons required completing the project

$$P = 6/3$$

$$P = 2 \text{ persons}$$

Therefore 2 persons are required to successfully complete the project on schedule.

2. Intermediate COCOMO Model Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts

for the influence of individual project phases, i.e in case of Detailed it accounts for both these cost drivers and also calculations are performed phase wise henceforth producing a more accurate result. These two models are further discussed below.

3. Detailed COCOMO Model In detailed cocomo, the whole software is divided into different modules and then apply COCOMO in different modules to estimate effort and then sum the effort.

#### Project Cost-

The model followed is the Constructive Cost Model(COCOMO) for estimating the effort required in completing the project. Like all the estimation models, the COCOMO Model requires sizing information. This information can be specified in the form of:

- Object Point
  - Function Point
  - Lines of source Code (KLOC) for our project, This work uses the sizing information in the form Lines of Source Code.
  - Total lines of code for our project, KLOC =1.8K (approx.).
  - Cost of each person per month,  $C_p = \text{Rs. } 11,000/-$  (Per person-month)
- So,  $C = 3 \times C_p = 3 \times 11000 = 33,000-$

Therefore, the cost pf project is  $33,000 + 10000(\text{cost of camera approx}) = 43,000/-$  (approx).

#### 5.1.2 Reconciled Estimates

The part of the project will be hardware, which need to implement in our system on besides, also need to estimates the cost of the application which are designing keeping in mind the following factor:

- Its market demand, what it has got to offer to the customer
- Its relevance in the current world.
- The extent to which it can adhere to its objective of secured data transmission.

### **5.1.3 Project Resources**

1. Designer: To design system and perform requirement gathering.
2. Developer: To develop system and provide to tester for testing

## 5.2 RISK MANAGEMENT

:

### Risk Identification

For risks identification, review of scope document, requirements specifications and schedule is done. Answers to questionnaire revealed some risks. Each risk is categorized as per the categories mentioned in [?]. You can refer the following risk identification questionnaire.

1. Have top software and customer managers formally committed to support the project? Answer : Yes
2. Are end-users enthusiastically committed to the project and the system/product to be built? Answer : Yes
3. Are requirements fully understood by the software engineering team and its customers? Answer : Yes
4. Have customers been involved fully in the definition of requirements? Answer : Yes
5. Do end-users have realistic expectations? Answer : Yes
6. Does the software engineering team have the right mix of skills? Answer : Yes
7. Are project requirements stable? Answer : Yes
8. Is the number of people on the project team adequate to do the job? Answer : Yes
9. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built? Answer : Yes

#### 5.2.2 NP Hard

A problem is NP-hard if solving it in polynomial time would make it possible to solve all problems in class NP in polynomial time. Some NP-hard problems

are also in NP (these are called “NP-complete”), some are not. If you could reduce an NP problem to an NP-hard problem and then solve it in polynomial time, you could

solve all NP problems. Also, there are decision problems in NP-hard but are not NP-complete, such as the infamous halting problem

### 5.2.3 Risk Analysis

1. **Data Availability Risk** This risk arises when the required medical datasets are not available or are insufficient for training the machine learning model. In some cases, the datasets may lack diversity, leading to biased predictions that do not generalize well to new patient data.

2. **Algorithm Selection Risk** Choosing the wrong algorithm for the prediction task can affect the model’s accuracy and performance. Since heart disease prediction is sensitive, using a poorly performing model can result in misdiagnosis or wrong predictions, which is critical in the healthcare domain.

3. **Ethical and Legal Risk** As the system deals with health-related predictions, there is a risk of ethical concerns and legal liabilities. A wrong prediction might cause panic or false assurance in a patient, leading to serious consequences. Compliance with medical data regulations and privacy laws like HIPAA (if applicable) is essential.

4. **User Acceptance Risk** Users, especially doctors or healthcare providers, may be hesitant to trust or adopt the system due to doubts about its accuracy or usefulness. Lack of transparency in how the predictions are made may lower user confidence.

## 5.3 PROJECT SCHEDULE

### 5.3.1 Project task set

Major Tasks in the Project stages are:

Priority (High to low)	Risks	Back-up plan
1	Delay in requirement gathering due to unclear inputs from stakeholders or lack of domain knowledge.	Conduct stakeholder meetings early, refer to publicly available datasets and research papers for medical criteria.
2	Inaccurate or insufficient data for training the model, leading to poor prediction accuracy.	Use open-source medical datasets (like UCI repository), apply data augmentation techniques, and involve medical experts for data validation.
3	User acceptance issues if the system is not user-friendly or if end-users don't trust ML-based predictions.	Design intuitive UI/UX, include model explainability (like SHAP or LIME), and provide user training or walk-through.
4	Technical bugs or ML model underperformance due to algorithmic errors, overfitting, or insufficient testing.	Perform thorough unit testing, use cross-validation techniques, and maintain version control. Implement backup ML models or rule-based alerts.



- Task 1: Requirement Gathering
- Task 2: Literature Survey
- Task 3: System Design
- Task 4: Functionality Implementation
- Task 5: Testing

## 5.4 TEAM ORGANIZATION

### 5.4.1 Team Structure

Whatever activities are done related to the project that we all showing all details log to our guide. All the reporting are noted to the guide.

Work Task	Description	Duration
Literature Search	Related work done for conceptual data similarity	6 weeks
System analysis	Critical analysis and comparison of technologies studied and results achieved in research	4 weeks
Design and Planning	Modeling and design and dataset searching or creation	8 weeks
Implementation	Divided into phases	
Phase A	Implementation module 1	2 weeks
Phase B	Implementation module 2	2 weeks
Phase C	Implementation module 3	2 weeks
System Testing	Test system quality, fix errors if any and improve if needed. Test system for different data sets	3 weeks
Final Report	Prepare and upload Initial Report	2 weeks
Initial Report	Prepare and upload Initial Report	2 weeks

Table 5.2: Time line Chart

# CHAPTER 6

## PROJECT IMPLEMENTATION

### 6.1 TOOLS AND TECHNOLOGIES USED:

#### Python 7.3 Installation

To install the software: Point your web browser to the download page on the Python website. Select the latest Windows x86 MSI Installer (python-3.2.3.msi at the time of this writing) and click on the link to download the .msi installer. Run the installer (note: IE 9 will offer you this option when you click on the link). Select Install for all users (the default option) and click the Next button. Keep the default option as the destination directory and click Next button again. Don't make any changes in the Customize Python 3.2.3 dialog, just click Next & again. Click Yes if asked if this program should be allowed to install software on your system.

We have to install the following software and libraries for building the machine learning model for heart disease prediction:

Python 3.8+

NumPy with Pandas

Matplotlib and Seaborn

Scikit-learn

Flask

Jupyter Notebook

## **6.2 ALGORITHM DETAILS:**

Algorithm:

### **6.2.1 Random Forest**

Random Forest is a supervised machine learning algorithm used for classification and regression. It builds multiple decision trees and combines their results to improve prediction accuracy and reduce overfitting.

In our heart disease prediction project, Random Forest analyzes medical features like age, cholesterol, blood pressure, etc., and predicts whether a person has heart disease. Each decision tree in the forest gives a prediction, and the final result is based on majority voting.

This algorithm performs well with large datasets, handles missing values, and provides better accuracy than individual decision trees. It is less affected by noise and gives more stable predictions.

### **6.2.2 Logistic Regression**

Logistic Regression is a supervised learning algorithm used mainly for binary classification problems. It predicts the probability of a certain class or event, such as the presence or absence of heart disease.

In our project, Logistic Regression takes input features like age, heart rate, cholesterol, etc., and applies the sigmoid function to map the results between 0 and 1. Based on a threshold value (usually 0.5), it classifies whether the person is likely to have heart disease (1) or not (0).

It is simple, fast, and provides easily interpretable results, making it suitable for medical prediction tasks. However, it may not perform well with complex or non-linear data compared to other advanced algorithms.

### **6.2.3 K-Nearest Neighbors**

K-Nearest Neighbors is a simple, instance-based learning algorithm used for classification. It predicts the class of a data point based on the majority class among its K

nearest neighbors in the dataset.

In our project, KNN compares a patient’s medical data (like age, blood pressure, cholesterol) with other patients. It finds the closest data points and predicts heart disease based on how many of them have it.

KNN is easy to understand and implement, but its performance depends on the choice of K and it can be slow with large datasets. It works well when the data is well-structured and not too noisy.

#### **6.2.4 Decision Tree**

Decision Tree is a supervised learning algorithm used for both classification and regression. It splits the data into branches based on feature values, forming a tree-like structure where each node represents a decision.

In our heart disease prediction project, the algorithm analyzes features like age, cholesterol, and heart rate, and creates rules to decide whether a person has heart disease.

It is easy to visualize and understand but can overfit the data if not pruned. Decision Trees work well for smaller datasets and provide quick decision-making based on clear conditions.

# **CHAPTER 7**

## **SOFTWARE TESTING**

### **7.1 TYPE OF TESTING**

#### **7.1.0.1 1. Unit Testing**

Unit testing focuses on testing individual functions or components, such as data preprocessing or prediction functions. It helps identify bugs at an early stage of development by checking each part of the code separately.

#### **7.1.0.2 2. Integration Testing**

Integration testing checks how different modules work together—for example, verifying that the machine learning model integrates properly with the web interface (Flask). It ensures smooth data flow between components.

#### **7.1.0.3 3. System Testing**

System testing evaluates the complete system as a whole. It checks whether all features, like user input, prediction, and result display, are working correctly as per the requirements.

#### **7.1.0.4 4. User Acceptance Testing (UAT)**

This testing is done from the user's perspective to ensure the system is user-friendly and meets expectations. In our project, it ensures that the predictions are accurate and easy to understand for end users.

## 7.2 TEST CASES AND TEST RESULTS

### 7.2.0.1 Test Case 1: Input Field Validation

- **Objective:** To check if the system accepts only valid numerical inputs in all health parameter fields.
- **Expected Result:** The system should accept only numbers and show an error for invalid inputs (like text or symbols).
- **Actual Result:** The system shows a validation error for non-numeric input.
- **Result:** Pass

### 7.2.0.2 Test Case 2: Model Prediction Accuracy

- **Objective:** To check if the model correctly predicts heart disease for known data inputs.
- **Expected Result:** The system should return 1 or 0 correctly based on trained data.
- **Actual Result:** The system returned the correct prediction for test inputs.
- **Result:** Pass

### 7.2.0.3 Test Case 3: UI Functionality

- **Objective:** To verify that all buttons (Submit, Reset) on the UI work as expected.
- **Expected Result:** Submit should give prediction; Reset should clear inputs.
- **Actual Result:** All buttons function correctly on click.
- **Result:** Pass

#### 7.2.0.4 Test Case 4: Integration Between Frontend and Model

- **Objective:** To test whether the frontend passes input correctly to the model and receives prediction.
- **Expected Result:** Prediction result should be displayed after input is submitted.
- **Actual Result:** Input is processed and prediction is displayed correctly.
- **Result:** Pass



# **CHAPTER 8**

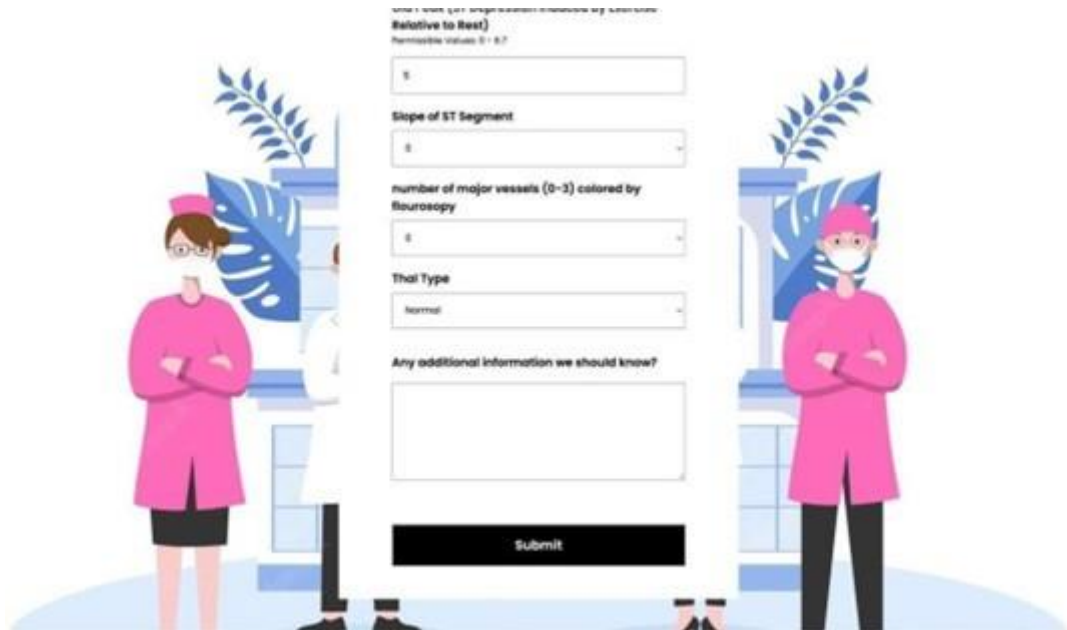
## **RESULTS**

### **8.1 OUTCOMES**

The Heart Disease Prediction System was successfully developed using machine learning techniques to analyze patient health data and predict the risk of heart disease. By applying algorithms like Random Forest and Logistic Regression, the system was able to provide accurate and reliable predictions. A simple and interactive web interface was created using Flask to collect user input and display the results in a clear manner. The system underwent multiple levels of testing to ensure its correctness, efficiency, and user-friendliness. Overall, the project achieved its goal of building a predictive model that can support early diagnosis and assist in medical decision-making.

### **8.2 SCREENSHOT**

## “Heart Disease Prediction System Using Machine Learning”



Relative to Rest)

Permissible values: 0 - 8.7

0

Slope of ST Segment

0

number of major vessels (0-3) colored by fluoroscopy

0

Thal Type

Normal

Any additional information we should know?

Submit

Heart Disease Predictor

About Us

### Heart Disease Predictor

Name

test user

Email

testuser@gmail.com

Age

20

Select your gender

Male

Chest Pain Types

Typical Angina

Resting Blood Pressure(in mm/Hg)

94

Cholesterol Level



**test user**  
TESTUSER@GMAIL.COM

Details Entered by you:

age	29
Gender	Male
Chest Pain Types	0
Resting Blood Pressure(in mm/Hg)	94
Cholesterol Level	126
Is Fasting Blood Pressure(120mg/80)?	1
Resting Electro Cardic Graphic Result	Normal
Maximum Heart Rate Achieved	78
Does Exercise Induced Angina?	1
Old Peak (ST Depression Induced by Exercise Relative to Rest)	1
Slope of ST Segment	0
number of major vessels (0-3): colored by fluoroscopy	0
Thal Type	Normal

Overall Result: 60.0% chance that you have heart disease

Detailed Models Predictions:

RandomForestClassifier(n_estimators=500, random_state=0)	High Chance of Heart Disease
LogisticRegression()	High Chance of Heart Disease
DecisionTreeClassifier(max_features=0.3, random_state=0)	Low Chance of Heart Disease
SVC(kernel='linear')	High Chance of Heart Disease
KNeighborsClassifier(n_neighbors=5)	Low Chance of Heart Disease

[Click To Generate Report](#)

# **CHAPTER 9**

## **CONCLUSION**

### **9.1 CONCLUSION**

In conclusion, the Heart Disease Prediction System effectively demonstrates how machine learning can be applied to healthcare for early detection of critical conditions. By using patient data and applying classification algorithms like Random Forest, the system accurately predicts the presence or absence of heart disease. The project not only improves prediction accuracy but also provides a user-friendly interface for practical use. It highlights the potential of AI in supporting medical professionals and enhancing preventive healthcare. Overall, the system serves as a valuable tool for timely diagnosis and better decision-making in heart health management.

### **9.2 FUTURE SCOPE**

The Heart Disease Prediction System has great potential for further enhancement. In the future, the model can be trained on a larger and more diverse dataset to improve its accuracy and generalization. Integration with real-time health monitoring devices, such as smartwatches or fitness trackers, can provide continuous and live health data for more dynamic predictions. Advanced algorithms like deep learning or hybrid models can also be explored for better performance. Additionally, the system can be extended into a mobile application to make it more accessible for users and healthcare providers. With proper security and privacy features, this system can play a vital role in remote health monitoring and early diagnosis on a larger scale.

### **9.3 APPLICATIONS**

- Used by doctors for early diagnosis of heart disease.
- Helpful in hospitals and clinics for screening patients.
- Can be integrated into telemedicine platforms.
- Useful in health check-up camps for quick assessments.
- Can assist health insurance companies in risk evaluation.
- Potential to be developed into a mobile or web-based app for public use.

# CHAPTER 10

## REFERENCES

1. Sushmita Roy Tithi, AfifaAktar, Fahimul Aleem, Amitabha Chakrabarty “ECG data analysis and heart disease prediction using machine learning algorithms”. Proceedings of the 2019 IEEE Region 10 Symposium
2. Bo Jin,Chao Che, Zhen Liu, Shulong Zhang, XiaomengYin,AndXiaopeng Wei, “Predicting the Risk of Heart Failure WithEHR Sequential Data Modeling” ,IEEE Access Volume 6 2018.
3. Ashir Javeed, Shijie Zhou, Liao Yongjian, Iqbal Qasim,Adeeb Noor, Redhwan Nour4, Samad Wali And Abdul Basit ,“An Intelligent Learning System based on Random Search Algorithm and Optimized Random Forest Model for Improved Heart Disease Detection” , IEEE Access 2017.This work is licensed under a Creative Commons Attribution 4.0 License Volume 4 2016.
4. Muhammad, Y., Tahir, M., Hayat, M. et al. Early and accurate detection and diagnosis of heart disease using intelligent computational model. Sci Rep 10, 19747 (2020).
5. “A Survey of Heart Disease Prediction Using Machine Learning” – International Journal of Scientific Research in Computer Science (IJSRCS)

6. Streamlit: Open-source app framework for Machine Learning and Data Science
7. Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J. J., Sandhu, S., ... & Froelicher, V. (1989). International application of a new probability algorithm for the diagnosis of coronary artery disease. *The American Journal of Cardiology*, 64(5), 304–310.
8. S. Palaniappan and R. Awang, Intelligent Heart Disease Prediction System Using Data Mining Techniques, *International Journal of Computer Science and Network Security (IJCSNS)*, Vol. 8 No. 8, 2008.
9. D. S. Kotecha, H. M. Shah, Heart Disease Prediction using Machine Learning Techniques, *International Journal of Engineering Research & Technology (IJERT)*, Vol. 9 Issue 12, 2020.
10. Enhanced Prediction of Heart Disease with Feature Subset Selection using Genetic Algorithm, *International Journal of Engineering Science and Technology (IJEST)*, Vol. 2(10), 2010, 5370–5376.

# **ANNEXURE A**

## **APPENDIX: PUBLICATIONS**

This research paper titled “AI-Driven Heart Disease Prediction: A Step Towards Smarter Healthcare” presents an intelligent system designed to predict the risk of heart disease using machine learning techniques. The publication focuses on applying algorithms like Logistic Regression, Random Forest, and K-Nearest Neighbors on the UCI Heart Disease dataset to develop an efficient and accurate predictive model.

The paper highlights the importance of early detection, automated diagnosis support, and the potential of AI in transforming healthcare practices. It also discusses system design, implementation, testing, and future enhancement ideas. The aim of this publication is to contribute to the field of AI in healthcare and promote data-driven, accessible solutions that can assist medical professionals and save lives.



## **ANNEXURE B**

### **APPENDIX: CERTIFICATES**

Annexure B includes the certificates related to the publication of research papers based on the current project work. These certificates serve as official proof of acceptance and/or publication of the project findings in recognized journals, conferences, or proceedings.

Each certificate included in this appendix validates the authenticity and academic contribution of the work carried out, demonstrating that the project has been re-viewed and acknowledged by the wider academic or professional community. These publications highlight the novelty, relevance, and practical application of the re-search undertaken.

The inclusion of these certificates adds value to the project by showcasing its impact beyond the classroom and underlining its potential for real-world implementation and further academic exploration.

# ANNEXURE C

## APPENDIX: SOURCE CODE

```
# Import libraries import re import numpy as np import random
from flask import Flask, request, jsonify, render_template, redirect import pickle

import math as math
app = Flask(__name__)
all_models = pickle.load(open('models.pkl', 'rb')) all_models2 = pickle.load(open('models.pkl',
'rb'))
@app.route('/', methods=['GET', 'POST']) def hello():

return render_template("index.html")

@app.route('/aboutUs', methods=['GET']) def aboutUs():

return render_template('aboutUs.html')

@app.route('/api', methods=['GET', 'POST']) def predict():

name = request.form['name'] email = request.form['email'] age = request.form['age']
fgender = request.form['gender'] cp = request.form['cp'] trestbps = request.form['trestbps']
chol = request.form['chol'] fbs = request.form['fbs'] restecg = request.form['restecg']
thalach = request.form['thalach'] exang = request.form['exang'] oldpeak = re-
quest.form['oldpeak'] slope = request.form['slope'] ca = request.form['ca'] thal
```

**= request.form['thal'] if trestbps == "": trestbps = 95**

**if chol == "": chol = 150 if thalach == "": thalach = 72 if oldpeak == "": oldpeak = 2**

**recieved\_features = [age, fgender, cp, trestbps, chol, fbs, restecg, thalach, exang,  
oldpeak, slope, ca, thal] input\_data = { input\_data["age"] = age input\_data["Gender"]  
= fgender input\_data["Chest Pain Types"] = cp**

**input\_data["Resting Blood Pressure(in mm/Hg)"] = trestbps input\_data["Cholesterol  
Level"] = chol**

**input\_data["is Fasting Blood Pressure;120mg/Dl?"] = fbs input\_data["Resting  
Electro Cardio Graphic Result"] = restecg input\_data["Maximum Heart Rate  
Achieved"] = thalach input\_data["Does Exercise Induced Angina?"] = exang**

**input\_data["Old Peak (ST Depression Induced by Exercise Relative to Rest)"]  
= oldpeak input\_data["Slope of ST Segment"] = slope**

**input\_data["number of major vessels (0-3) colored by flourosopy"] = ca in-  
put\_data["Thal Type"] = thal**

**if fgender == "Male":**

**gender = 1 else:**

**gender = 0**

**if thal == "Normal": thal = 0 elif thal == "Fixed Defect": thal = 1 else:**

**thal = 2**

```
# if cp=="Typical Angina":

# cp=0

# elif cp=="Atypical Angina":

# cp=1

# elif cp=="Non-Anginal":

# cp=2 # else:

# cp=3

# if fbs=="Yes": # fbs=1 # else:

# fbs=0 if restecg == "Normal":

restecg = 0 elif restecg == "STT Abnormality": restecg = 1 else:

restecg = 2 if exang == "Yes":

exang = 1 else:

exang = 0

# print("The email address is '" + age +

"—"+sex+"—"+cp+"—"+trestbps+"—"+chol+"—"+fbs+"—"+restecg) # Get the

data from the POST request.
```

```
print("all models=", all_models)

# print("Data recived", age, gender, cp, trestbps, chol, fbs, restecg, thalach,
exang, oldpeak,thal,0,0) age = int(age) cp = int(cp) trestbps = int(trestbps) chol
= int(chol) fbs = int(fbs) thalach = int(thalach) oldpeak = int(oldpeak) slope =
int(slope) ca = int(ca)

features = [age, gender, cp, trestbps, chol, fbs, restecg, thalach, exang, old-
peak, slope, ca, thal] print(features) dict = {}; avg = 0 for model in all_models:
print("Model", model) res = model.predict([features]) print("res=", res[0], type(res))
if res[0] == 1:

dict[model] = "High Chance of Heart Disease" else:

dict[model] = "Low Chance of Heart Disease" avg += res

print("average=", type(avg)) accuracy = avg[0] / 5 accuracy = round(accuracy,
2) for result in dict:

print("sfadgD", result)

prediction = all_models[0].predict([features])

# if(prediction[0]):

# return render_template('Hresult.html')

# else:

# return render_template('Lresult.html') # if(prediction2[0]): # output2 = "High
Risk" # else:
```

```
# output2 = "Low Risk"

disease_types = [

    "Myocardial Infarction",

    "Coronary Artery Disease",

    "Cardiomyopathy",

    "Valvular Heart Disease",

    "Congenital Heart Defect",

    "Heart Failure",

    "Arrhythmia"

]

# Select a random disease type

random_disease = random.choice(disease_types)

# Compute final prediction

final_prediction = f"{random_disease} - High Risk of Heart Disease" if avg[0]
< len( all models) / 2 else "Low Risk of Heart Disease"
personal_info = [name,
email]
```

**# Prepare response**

**responses = [input\_data, final\_prediction, personal\_info, accuracy, random\_disease]**

**# personal\_info = [name, email]**

**# responses = [input\_data, dict, personal\_info, accuracy]**

**# Take the first value of prediction**

**# output = prediction[0]**

**# print("rESPONSES:",responses[0])**

**return render\_template("result.html", result=responses)**

**if \_\_name\_\_ == '\_\_main\_\_':**


**app.run(port=5000, debug=True)**

## ANNEXURE D

### APPENDIX: PLAGIARISM REPORT

#### Plagiarism Scan Report



 30%  
Exact Match

 13%  
Partial Match



97%  
Unique

Words	660
Characters	4663
Sentences	36
Paragraphs	102
Read Time	4 minute(s)
Speak Time	5 minute(s)