
HEART DISEASE PREDICTION SYSTEM

Project Report

SUPERVISOR: Prof. Vibha Gaur
Mrs. Vandita Grover
Miss. Nishu Singh

SUBMITTED BY

Keshav Gaur 20001570026
Deepanshu Kumar 20001570017



2022

Department of Computer Science

ACHARYA NARENDRA DEV COLLEGE

ACKNOWLEDGEMENT

This Project was jointly undertaken by Keshav Gaur and Deepanshu Kumar as their Semester-IV Software Engineering Project, under the guidance and supervision of Prof. Vibha Gaur, Mrs. Vandita Grover and Miss Nishu Singh. Our primary thanks goes to her, who poured over every inch of our project with painstaking attention and helped us throughout the working of the project. It is our privilege to acknowledge our deepest gratitude to her for her inspiration which has helped us immensely. We are extremely grateful to her for her unstilted support and encouragement in the preparation of this project.

Keshav Gaur

Deepanshu Kumar

ACHARYA NARENDRA DEV COLLEGE
(University of Delhi)

CERTIFICATE

This is to certify that the project entitled “Heart Disease Prediction System” has been done by: Keshav Gaur and Deepanshu Kumar of Bachelor of Computer Science (Hons.) during Semester-IV from Acharya Narendra Dev College, University of Delhi under the supervision of Prof. Vibha Gaur, Mrs Vandita Grover and Miss Nishu Singh .

Keshav Gaur

Deepanshu Kumar

Supervisor

Prof. Vibha Gaur

Mrs. Vandita Grover

Miss Nishu singh

Contents	Page no.
1 PROBLEM STATEMENT	6
2 PROCESS MODEL	7
3 REQUIREMENT ANALYSIS & MODELING	8 - 16
3.1 DFD	
3.1.1 Context Level DFD	
3.1.2 Level 1 DFD	
3.2 Data Dictionary	
3.3 Use Case Diagram	
3.4 Sequence Diagram	
4 SOFTWARE REQUIREMENT SPECIFICATION (SRS)	17 - 19
4.1 Overall Description	
4.1.1 Product Functions	
4.1.2 User Characteristics	
4.1.3 General Constraints	
4.1.4 Assumptions and Dependencies	
4.2 External Interface Requirements	
4.2.1 User Interface	
4.2.2 Hardware Interface	
4.2.3 Software Interface	
4.3 Functional requirements	
4.3.1 FR1 Login Requirement	
4.3.2 FR2 Input Requirement	
4.3.3 FR3 Model Training Requirement	
4.4 Performance Requirements	
4.4.1 Speed	
4.4.2 Response Time	
4.5 Design Constraints	
5 ESTIMATIONS	20 - 23
5.1 Function Points	
5.2 Effort	
5.3 Cost	

6 SCHEDULING	24
6.1 Timeline Chart	
7 RISK MANAGEMENT	25
7.1 Risk Table	
8 DESIGN	26
8.1 Structural Chart	
8.2 Pseudo Code	
9 CODING	27 - 29
9.1 Coding of Prediction Module	
9.2 Classification Model Used	
9.3 Output of Prediction Module	
10 TESTING	30-33
10.1 Test Case Design	
10.2 Flow Graph	
10.3 Basis Path Set	
10.4 Cyclomatic Complexity.	
11 REFERENCES	34

Chapter 1

PROBLEM STATEMENT

Heart disease is one of the most significant causes of mortality in the world today. Prediction of cardiovascular disease is a critical challenge in the area of clinical data analysis. The model is proposing a novel method that aims at finding significant features by applying machine learning techniques resulting in improving the accuracy in the prediction of cardiovascular disease.

To predict the heart disease model is using a dataset which is a combination of 4 different databases, but only the UCI Cleveland dataset was used. This database consists of a total of 76 attributes but all published experiments refer to using a subset of only 12 features. A lot of work has been carried out to predict heart disease using the UCI Machine Learning dataset. Different levels of accuracy have been attained using various data mining techniques [3].

Chapter 2

PROCESS MODEL

A software process model is an abstraction of the software development process. The models specify the stages and order of a process.

The waterfall model

The waterfall model, often known as the classic life cycle, proposes a systematic, sequential approach to software development that starts with client requirements and goes through planning, modeling, construction, and deployment, concluding with the continuing support of the completed software (Figure1).

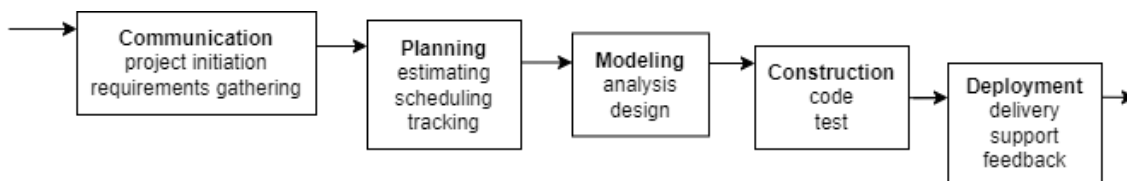


Figure1 The waterfall model [1]

The waterfall model depicts the software development process as a sequential flow of events. This indicates that any step of the development process can start only after the previous one has finished. The phases in this waterfall model do not overlap. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The model is used for this software project because the project aims to predict the condition of the heart with the help of some attributes. All requirements are very well known, clear, and fixed. Technology to be used is understood. There are no ambiguous requirements. Ample resources with the required expertise are available freely. The project is short and focuses only on a specific task, i.e., accurate prediction of heart disease.

Chapter 3

REQUIREMENT ANALYSIS & MODELLING

3.1 DFD

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself.

3.1.1 Context Level DFD

It is a basic overview of the whole system or process being analyzed or modeled. It is designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities.

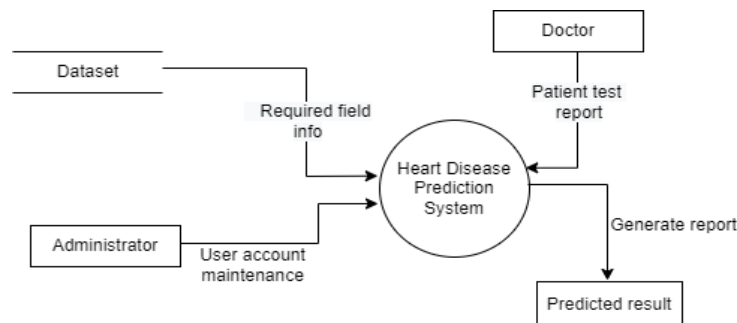


Figure 3.1: Context Level DFD

3.1.2 Level 1 DFD

A level 1 DFD notates each of the main sub-processes that together form the complete system. It is an ‘exploded view’ of the context diagram.

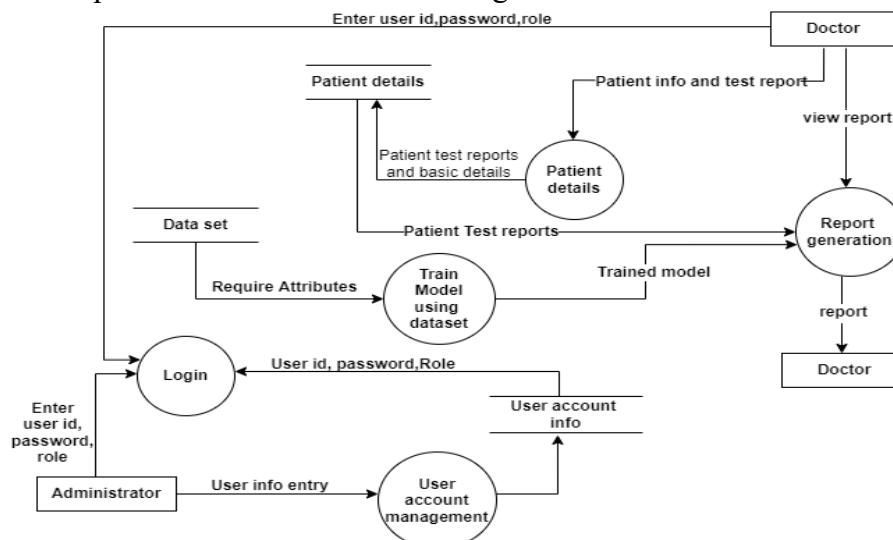


Figure 3.2: Level 1 DFD

3.2 Data Dictionaries

Data Dictionary is the major component in the structured analysis model of the system. A data dictionary in Software Engineering means a file or a set of files that includes a database's metadata (hold records about other objects in the database), like data ownership, relationships of the data to another object, and some other data.

Components of Data Dictionary:

In Software Engineering, the data dictionary contains the following information as follows.

Name of the item – It can be your choice.

Aliases – It represents another name.

Description – Description of what actual text is all about.

Related data items – Relationship with other data items.

Range of values – It will represent all possible answers

Table 3.1: Data dictionary notation and meaning

Notation	Data Type	Meaning
User info entry	String	User id ,password and role of the user
User id	String	User id to log in to the system
Password	String	User password to login to the system
Role	String	User role that is login to the system [Admin / Doctor]
Patient info/basic details	String, Integer	Basic details of the patient like Patient's id, Name, Date Of Birth, Sex, Contact No., Date of check up
Require Attributes	Float	Entries of the following attributes from the dataset: ChestPainType, RestingBP, Cholesterol, FastingBS, RestingECG, MaxHR, ExerciseAngina, Oldpeak, ST_Slope, Heart Disease
Patient Test reports	Float	Values of the following attributes : ChestPainType, RestingBP , Cholesterol, FastingBS, RestingECG, MaxHR, ExerciseAngina, Oldpeak, ST_Slope
Report	String	Patient's heart report showing whether heart disease is present.

3.3 Use Case Diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well.

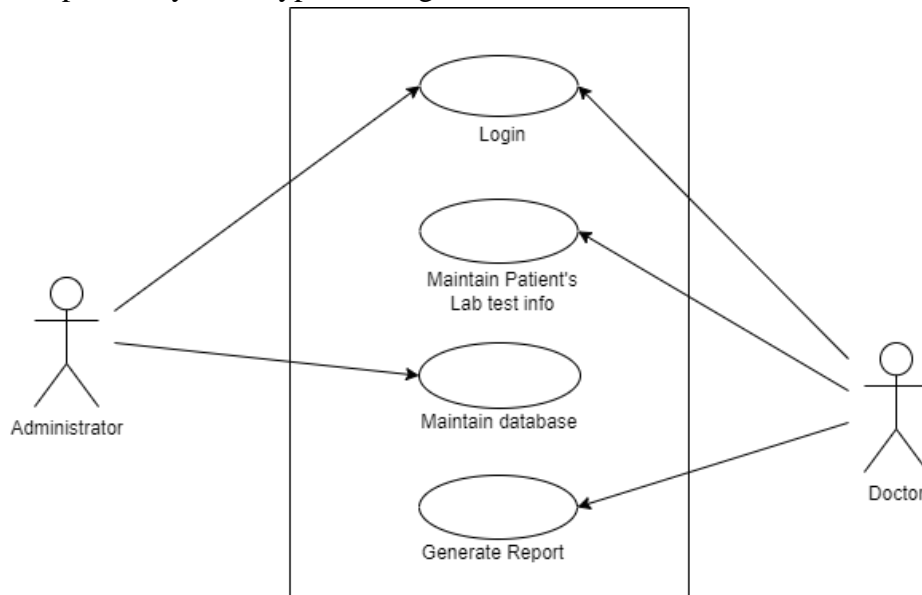


Figure 3.3: Use Case diagram [2]

Use Case Description

1. Login

1.1 Brief Description

This case describes how a user logs into the Heart Disease Prediction System.

1.2 Actors

The following actor(s) interact and participate in this use case:

Administrator and Doctor

1.3 Flow of Events

1.3.1 Basic Flow

This use case starts when the actor wishes to Login to the Heart Disease Prediction System.

1. The System requests that the actor enter his/her userid, password and role. The role can be either Doctor or Administrator.
2. The actor enters his/her userid, password and role.
3. The system validates the entered userid, password, role and login the actor into the system.

1.3.2 Alternative Flows

1.3.2.1 Invalid name/Password/Role

In the Basic Flow, the actor enters an invalid userid, password or role, the system displays an error message. The actor can choose to either return to the beginning of the Basic Flow or cancel the login, at which point the use case ends.

1.4 Special Requirements

No special requirements.

1.5 Pre-Conditions

All users must have a User Account (User ID, Password and Role) created for them in the system (through the Administrator), prior to executing the use cases.

1.6 Post-Conditions

If the use case was successful, the actor is logged into the system. If not, the system state is unchanged.

If the actor has role 'Doctor' he/she will have access to only screens corresponding to the Patient details, Lab test info maintenance and Report Generation modules of the System.

If the actor has role 'Administrator' he/she will have access to screens corresponding to the Database and Loginmaintenance modules of the system.

1.7 Extension Points

None

2 Maintain Patient Lab Test Info

2.1 Brief Description

This use case allows the actor with role 'Doctor' and 'Administrator' to maintain Patient Lab Test Info. This includes adding and updating its test information.

2.2 Actor(s)

The following actor(s) participate and interact in this use case:

Doctor and Administrator

2.3 Flow of Event

2.3.1 Basic Flow

This use case starts when the Doctor wishes to add Lab test Information from the system.

- 1) The system allows the doctor Add the test information only and the administrator can Add, Delete or Update a test information.
- 2) Once the Doctor provides the requested information, this sub-flow

is executed.

- If the Doctor selected 'Add a Lab test Information', the Add the Lab Test Information sub-flow is executed.
- 3) Once the Administrator provides the requested information, one of the sub flows is executed.
- Query entered by the administrator will be processed and will do the required changes in the database.

2.3.1.1 Add a test Information

- 1 The System requests that the Doctor enter the Patient's Lab Test Information. This includes:
 - Name
 - Age
 - Sex
 - Patient id
 - Test reports
- 2 Once the Doctor provides the requested information, the patient's lab test information will be added to the system and an appropriate message displayed.

2.3.1.2 Delete a test information

Administrator will write the query for a specific patient to delete his/her details. Then the system deletes that patient's record.

2.3.1.3 Update a test information

Administrator will write the query for a specific patient to update his/her details. Then the system updates that patient's record.

2.3.2 Alternative Flows

2.3.2.1 Patient not found

If in the Update a Lab Test Info Lab test info sub-flows, a patient with specific patient id does not exist, the system displays an error message. The Admin then can enter a different Patient id or cancel the operation at which point the use case end.

2.3 Special Requirements

Patient's Test must be done.

2.4 Pre-Conditions

The Doctor/Admin must be logged onto the system before this use case begins.

2.6 Post-Conditions

If the use case was successful, the patient's lab test information is added or updated to the database. Otherwise, the database state will remain unchanged.

2.7 Extension Points

None

3. Generate Report

3.1 Brief Description

This use case allows the actor with role 'Doctor' to generate reports. The following reports can be generated:

Patient Heart Disease Prediction Report

3.2 Actor

The following actor(s) interact and participate in this use case: Doctor

3.3 Flow of Events

3.3.1 Basic Flow

This use case starts when the Doctor wishes to generate reports.

3.3.1.1 The system requests the Doctor to specify the patient id for which he/she wants to generate report.

3.3.1.2 Once the Doctor provides the requested information , one of the sub-flows is executed:

- If Doctor selected "Generate Report", then Heart Disease Prediction Report sub-flow is executed.

3.3.1.3 If doctor forgot the patient id then he/she can find the patient's id by entering name of the patient and then click on find button.

3.3.2 Alternative Flows

3.3.2.1 Patient Not Found

If no patient information exists in the system for the patient id specified by the Doctor, the system displays an error message. The Doctor can then enter a different patient Id, can find patient id by entering the name of the patient or cancel the operation at which point the use case ends.

3.4 Special Requirements

None

3.5 Pre-conditions

The Doctor must be logged onto the system before this use case begins.

3.6 Post-Conditions

If the use case was successful, the desired report is generated. Otherwise the system state remains unchanged.

3.7 Extension points

None.

4. Maintain Database

4.1 Brief Description

This use case allows the actor with role ‘Administrator’ to maintain Database which consists of the Patient info and report. This includes adding, deleting and modifying its Database Information.

4.2 Actor(s)

The following actor(s) participate and interact in this use case:
Administrator

4.3 Flow of Events

4.3.1 Basic Flow

This use case starts when the Administrator wishes to add, delete and modify Lab test Information from the system.

The system requests that the Administrator to enter query to perform add, update or delete operation in the database.

4.3.2 Alternative Flows

If in the update a Patient’s Info, Report and delete a Patient’s info, a patient with specific patient id does not exist, the system displays an error message. The Administrator then can enter a different Patient id or cancel the operation at which point the use case ends.

4.4 Special Requirements

Patient Test Details and Report must be there.

4.5 Pre-Conditions

The Administrator must be logged onto the system before this use case begins.

4.6 Post-Conditions

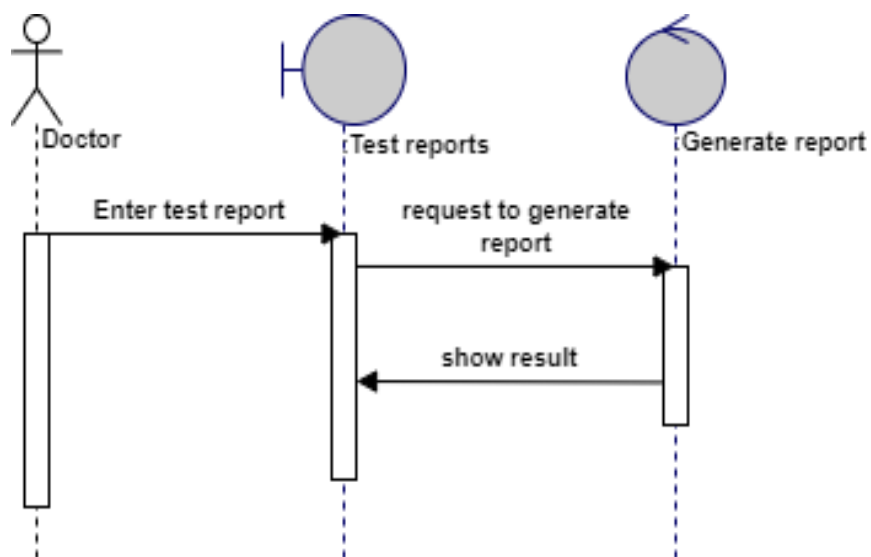
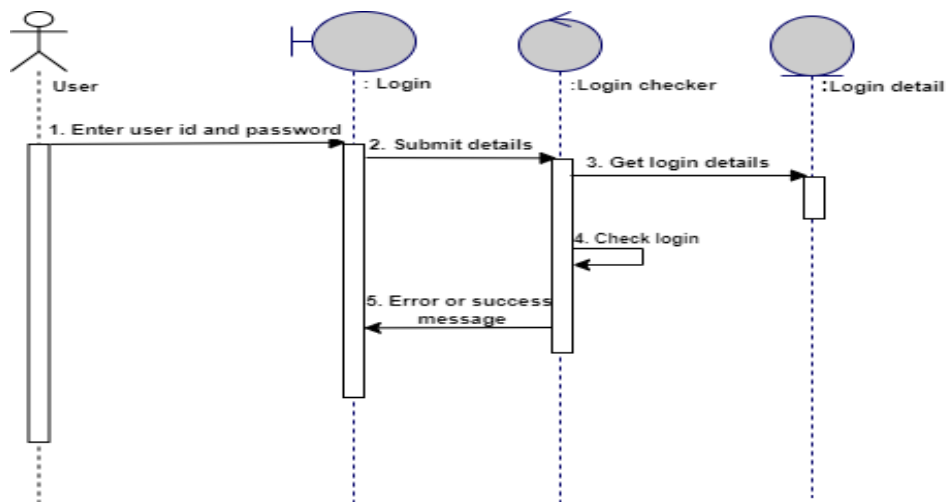
If the use case was successful, the patient's lab test information and report is added, deleted or updated in the database. Otherwise, the system state is unchanged.

4.7 Extension Points

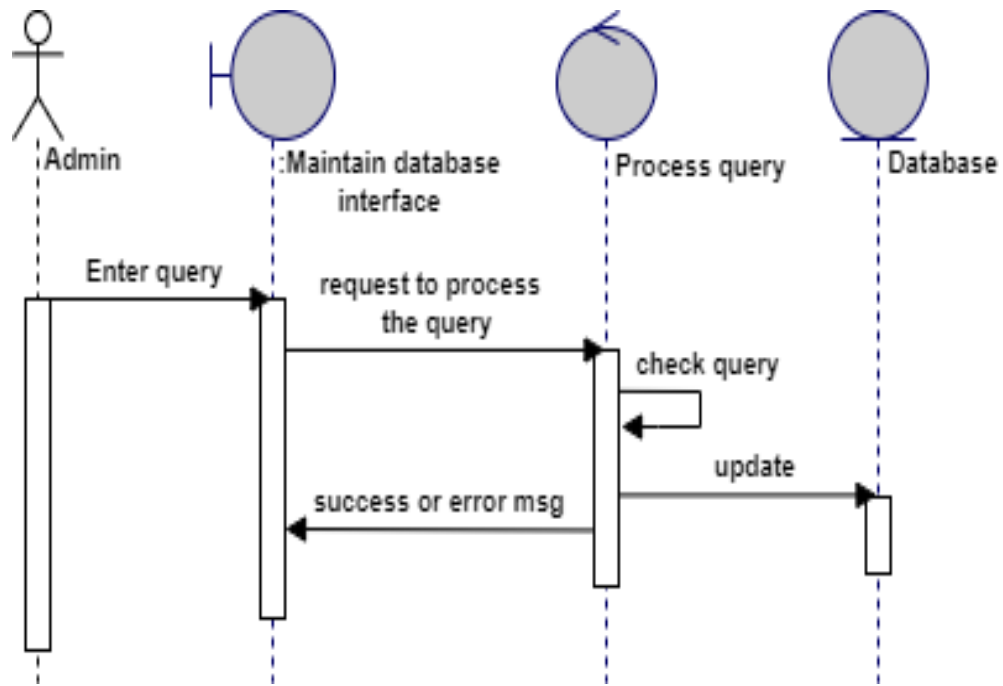
None

3.4 Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects work together.



Sequence diagram – generate report [2]



Sequence diagram - maintain database [2]

Chapter 4

SOFTWARE REQUIREMENT SPECIFICATION (SRS)

A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform. It also describes needs of all stakeholders (manager, developers, users) to fulfill.

4.1 Overall Description

The purpose of the Software Requirements Specification document is to clearly define the system under development, namely the Heart Disease Prediction System. The intended audience of this document includes Administrator of the Hospital, Doctor. Other intended audience includes the development team such as requirements team, requirement analyst, design team, and other members of the developing organization.

4.1.1 Product Functions

- When Doctor/Admin enters the System it pop ups login page where he/she enters his/her id, password and role.
- It is then validated by the system and proceeds to a menu page.
- For Doctor: Menu page contains 3 rows, first is of Patient information where doctor will enter patient's personal details and second is of lab test information where doctor will enter the lab test reports of the patient and third is of Report Generation where it asks for patient id and predict whether the patient is having any heart disease or not.
- For Admin: There will be an interface to write queries for changing/updating database.

4.1.2 User Characteristics

- The Doctor needs to be Keyboard literate and to able to use button, pull-down menus and similar tools.
- Admin needs to have knowledge of database management system.

4.1.3 Assumptions and Dependencies

- Doctor and Administrator should have the login id and password.
- The attribute values in the dataset must be accurate.
- All required details must be entered.
- Test Report data must be accurate.
- Patient must have lab reports.
- System using this software must have minimum 4GB RAM

4.2 External Interface Requirements

External interface are interfaces on which the product is dependent externally. For this software Dataset taken from 'Kaggle.com'.

4.3 Functional requirements

Functional requirements define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs.

4.3.1 FR1 Login Requirement

The system will allow access only to authorized users with specific roles (Administrator, Doctor). Depending upon the user's role, he/she will be able to access only specific modules of the system. For login into the system, the user has to enter user id, password, and role.

4.3.2 FR2 Input Requirement

The system will require inputs for eleven fields to predict the heart disease of the patient. The values for these fields will be taken from the user through a form interface and given to the model for performing further calculations to do prediction of the presence of the heart disease. Fields are Age, Sex, Chest pain type, Resting bp, Cholesterol, FastingBS, RestingECG, MaxHR, ExerciseAngina, Oldpeak, and ST slope.

4.3.3 FR3 Model Training Requirement

The system will require a dataset for training the model that will be predicting the heart condition. It is the most essential part of the system because the accuracy of the model depends upon the accuracy of the dataset.

4.4 Performance Requirements

Performance requirements typically comprise a set of criteria (speed, response time, recovery time) that stipulate how things should perform or the standards that they must achieve in a specific set of circumstances.

4.4.1 Speed

The performance of any system depends on the speed of that system. The system will took only 400 milliseconds (taken average) to predict heart disease.

4.4.2 Response time

The performance of a system also depends upon the response time i.e. the time taken by the system to produce the output. Response time of the system will be 100 milliseconds.

4.5 Design Constraints

Design constraints are conditions that need to happen for a project to be successful. Design constraints help narrow choices when creating a project.

Design constraints for the system are:

1. Software Language: All coding will be done in python 3.9.7.
2. Classification models: Logistic regression, Decision tree Classifier and KNN.
3. Only two persons (Administrator and Doctor) are allowed to interact with the system.

Chapter 5

ESTIMATIONS

5.1 Function Points

A Function point is a unit of measurement to express the amount of business functionality an information system provides to a user.

The function point (FP) metric can be used effectively as a means for measuring the functionality delivered by a system. Using historical data, the FP metric can then be used to (1) estimate the cost or effort required to design, code, and test the software; (2) predict the number of errors that will be encountered during testing; and (3) forecast the number of components and/or the number of projected source lines in the implemented system [1].

Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and qualitative assessments of software complexity. Information domain values are defined in the following manner:

Number of external inputs (EIs). Each external input originates from a user or is transmitted from another application and provides distinct application-oriented data or control information. Inputs are often used to update internal logical files (ILFs). Inputs should be distinguished from inquiries, which are counted separately.

Number of external outputs (EOs). Each external output is derived data within the application that provides information to the user. In this context external output refers to reports, screens, error messages, and the like.

Number of external inquiries (EQs). An external inquiry is defined as an online input that results in the generation of some immediate software response in the form of an online output (often retrieved from an ILF).

Number of internal logical files (ILFs). Each internal logical file is a logical grouping of data that resides within the application's boundary and is maintained via external inputs.

Number of external interface files (EIFs). Each external interface file is a logical grouping of data that resides external to the application but provides data that may be of use to the application.

Assuming weighting factor to be average for this project.

Table 5.1 Computing Count total [1]

Information Domain Value	Count	Weighting factor			Total
		Simple	Average	Complex	
External Inputs (EIs)	13	3	4	6	52
External Outputs (Eos)	1	4	5	7	5
External Inquiries (EQs)	0	3	4	6	0
Internal Logical Files (ILFs)	2	7	10	15	20
External Interface Files (EIFs)	0	5	7	10	0
Count total					77

To compute function points (FP), the following relationship is used:

$$FP = \text{count total} \times [0.65 + 0.01 \times \Sigma(F_i)]$$

The F_i ($i = 1$ to 14) are *value adjustment factors* (VAF) based on responses to the following questions:

Table 5.2: Computing function points [1]

S. No.	Questions	Points (F _i)
1.	Does the system require reliable backup and recovery?	2
2.	Are specialized data communications required to transfer information to or from the application?	0
3.	Are there distributed processing functions?	5
4.	Is performance critical?	5
5.	Will the system run in an existing, heavily utilized operational environment?	3
6.	Does the system require online data entry?	0
7.	Does the online data entry require the input transaction to be built over multiple screens or operations?	0
8.	Are the ILFs updated online?	0
9.	Are the inputs, outputs, files, or inquiries complex?	3
10.	Is the internal processing complex?	5
11.	Is the code designed to be reusable?	3
12.	Are conversion and installation included in the design?	0
13.	Is the system designed for multiple installations in different organizations?	3
14.	Is the application designed to facilitate change and ease of use by the user?	5

By answering to these questions we get the value of $\Sigma F_i = 34$. Therefore,

$$FP = 77 \times [0.65 + (0.01 \times 34)] = 76.23$$

5.2 Efforts

It is the process of forecasting how much effort is required to develop or maintain a software application. The effort is traditionally measured in the hours worked by a person, or the money needed to pay for this work.

Following steps are taken to estimate effort to develop a project

1. Access Object counts

Estimate the number of screens, reports and 3GL components that will compromise this application.

Number of screens = 6

Number of reports = 1

Number of 3GL components = 1

2. Assign complexity weights to each object

The weights are used for three object types, i.e., screens, reports and 3GL components. Complexity weight are assigned according to object's complexity level using following table

Table 5.3: Complexity weight table [5]

Object Type	Complexity Weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL Components	-	-	10

3. Determine Object points

Assuming complexity weight to be medium for this project.

Table 5.4: Calculate object points [5]

Object type	Number of object instances	Complexity weight	Object points
Screens	6	2	12
Reports	1	5	5
3GL Components	1	10	10

Total object point = $12 + 5 + 10 = 27$

4. Compute New Object Points (NOP)

We have to estimate the %reuse to be achieved in this project.

NOP are the object point that will need to be developed and differ from the object point count because there may be reuse of some object instance in a project.

$$\begin{aligned}\text{NOP} &= [(\text{object points}) * (100 - \% \text{reuse})] / 100 \\ &= [27 * (100 - 0)] / 100 = 27 \quad (\% \text{reuse} = 0)\end{aligned}$$

5. Calculate Productivity rate (PROD)

Productivity rate is calculated on the basis of information given about developer's experience and capability.

It can be calculated using the following table

Table: Productivity rate

Developers experience & capability	Productivity(PROD)
Very Low	4
Low	7
Nominal	13
High	25
Very High	50

Assuming experience and capability of developers to be Nominal, the productivity of the project is PROD = 13

6. Compute the estimated Effort

Effort to develop a project can be calculated as

$$\text{Effort} = \text{NOP} / \text{PROD}$$

Therefore,

$$\text{Effort} = 27/13 = 2.077 \text{ PM}$$

5.3 Cost

Cost to develop a project can be calculated as

$$\text{Cost} = \text{Effort} * \text{Burdened Rate}$$

Assuming Burdened Rate is 65,000 rupees per PM. Therefore

$$\begin{aligned}\text{Cost} &= 2.077 * 50000 \\ &= 1,35,005 \text{ rupees}\end{aligned}$$

Chapter 6

SCHEDULING

A Timeline is a chart which displays a project schedule in chronological order. A Timeline is used in project management to depict project milestones and visualize project phases, and show project progress.

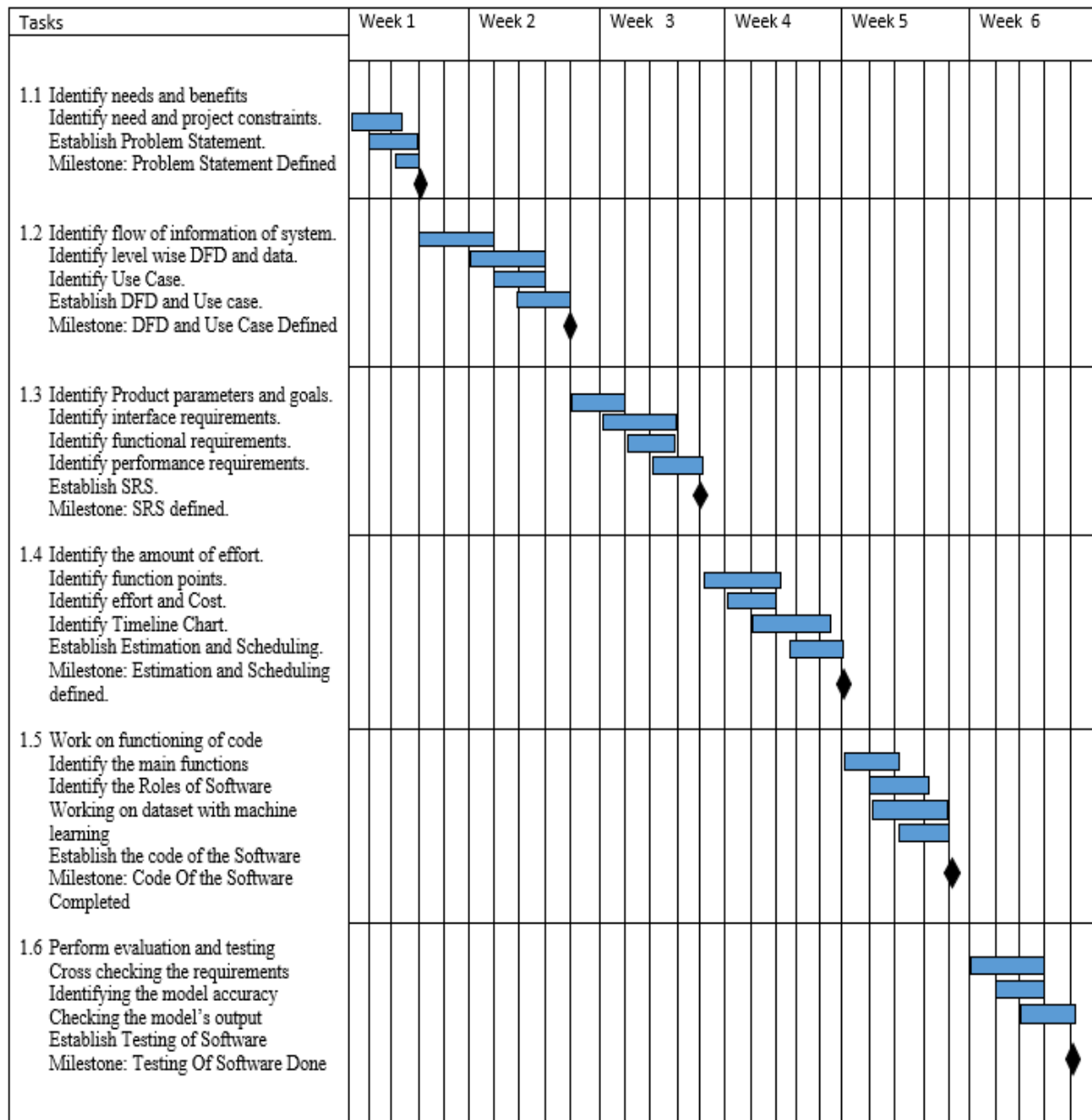


Figure 6.1 Time-line chart [2]

Chapter-7

RISK MANAGEMENT

Risk Table:

A risk assessment matrix (sometimes called a risk control matrix) is **a tool used during the risk assessment stage of project planning**. It's used to identify and capture the likelihood of project risks, as well as to evaluate the potential damage or interruption caused by those risks.

Risk table for our software is:

Table 7.1: Risk Table

Risk no	Problem	Category Of Risk	Probability of occurrence of Problem	Impact Of Risk	RMMM
R1	Checking accuracy with every model and choosing perfect model	CR	60%	2	Getting known to every model's risk and their working.
R2	Delivery deadline would be crossed	SR	40%	1	Working efficiently with team and taking time into consideration.
R3	Lack Of Training on Tools	PR	30%	2	Training team with tools beforehand.
R4	Staff Inexperienced	PR	40%	2	Making them work under experienced person
R5	Less Reuse than planned	SuR	60%	1	Having control on necessary modifications
R6	End users resist System	PR	40%	2	Making user write requirements properly and working on to fix bugs
R7	Large number of users than planned	PR	70%	2	Making model capable to increase memory than expected
R8	Inadequate Design-Redesign Required	SuR	30%	2	Working on Design with matching requirements
R9	Changes in the Requirements during Development	PR	40%	2	Making end user display the product as it proceeds ahead.

Impact Values: 1-Negligible
 2-Marginal
 3-Critical
 4-Catastrophic

Category of Risk: CR-Cost Risk
 SR-Schedule Risk
 PR-Performance Risk
 SuR-Support Risk

Chapter 8

DESIGN

Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation. Software design is the first step in SDLC (Software Design Life Cycle), which moves the concentration from problem domain to solution domain. It tries to specify how to fulfill the requirements mentioned in SRS.

8.1 Structure Chart

Structure Chart represent hierarchical structure of module. It breaks down the entire system into lowest functional modules. Structure Chart partitions the system into black boxes (functionality of the system is known to the users but inner details are unknown). Inputs are given to the black boxes and appropriate outputs are generated.

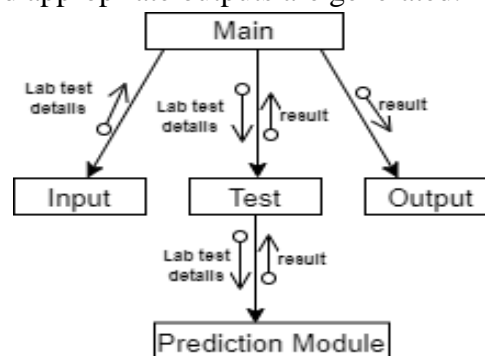


Figure 8.1: Structure Chart

8.2 Pseudo Code

It is a methodology that allows the programmer to represent the implementation of an algorithm.

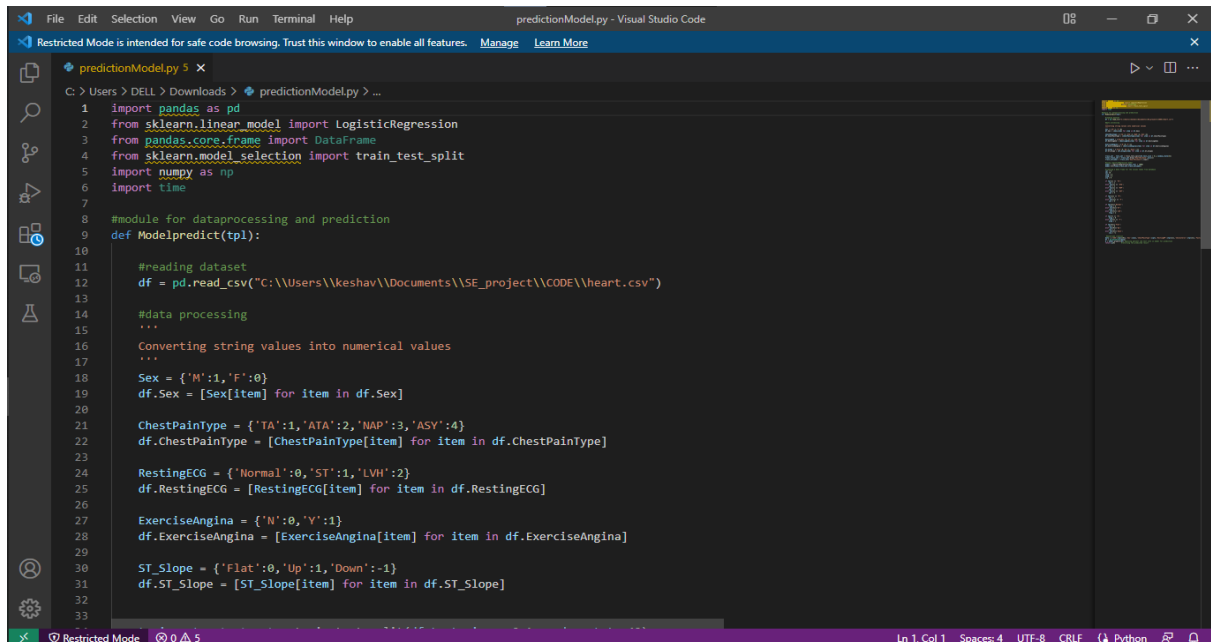
Pseudo code of the structure chart:

1. Begin
2. While user wants to continue:
 3. Take patient details from user
 4. Data pre-processing
 5. Model Training
 6. `result = Test()`
 7. If (`result == 1`)
 8. `print('Patient has heart disease')`
 9. else
 10. `print('Patient does not have heart disease')`
11. End if
12. End while
13. End

Chapter 9

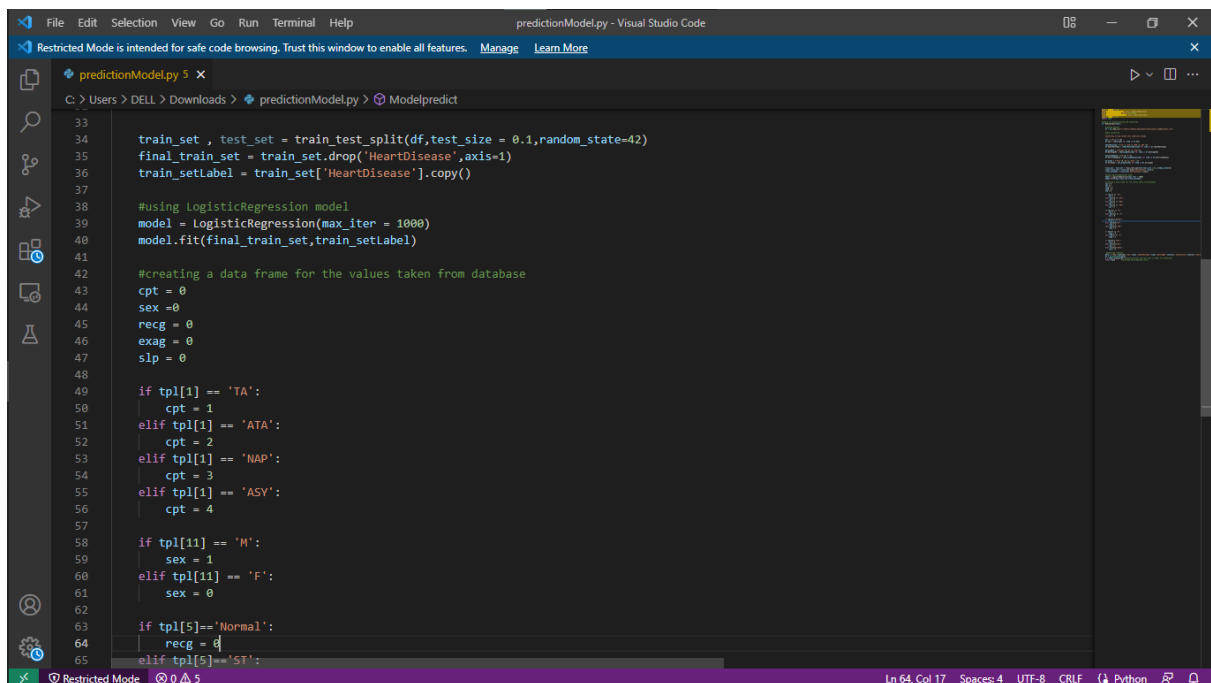
CODING

9.1 Coding of Prediction Module



```
File Edit Selection View Go Run Terminal Help predictionModel.py - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

predictionModel.py 5 X
C:\Users\DELL> Downloads > predictionModel.py > ...
1 import pandas as pd
2 from sklearn.linear_model import LogisticRegression
3 from pandas.core.frame import DataFrame
4 from sklearn.model_selection import train_test_split
5 import numpy as np
6 import time
7
8 #module for dataprocessing and prediction
9 def Modelpredict(tpl):
10
11     #reading dataset
12     df = pd.read_csv("C:\\Users\\keshav\\Documents\\SE_project\\CODE\\heart.csv")
13
14     #data processing
15     ...
16     Converting string values into numerical values
17     ...
18     Sex = {'M':1,'F':0}
19     df.Sex = [Sex[item] for item in df.Sex]
20
21     ChestPainType = {'TA':1,'ATA':2,'NAP':3,'ASY':4}
22     df.ChestPainType = [ChestPainType[item] for item in df.ChestPainType]
23
24     RestingECG = {'Normal':0,'ST':1,'LHV':2}
25     df.RestingECG = [RestingECG[item] for item in df.RestingECG]
26
27     ExerciseAngina = {'N':0,'Y':1}
28     df.ExerciseAngina = [ExerciseAngina[item] for item in df.ExerciseAngina]
29
30     ST_Slope = {'Flat':0,'Up':1,'Down':-1}
31     df.ST_Slope = [ST_Slope[item] for item in df.ST_Slope]
32
33
```



```
File Edit Selection View Go Run Terminal Help predictionModel.py - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

predictionModel.py 5 X
C:\Users\DELL> Downloads > predictionModel.py > Modelpredict
33
34 train_set , test_set = train_test_split(df,test_size = 0.1,random_state=42)
35 final_train_set = train_set.drop('HeartDisease',axis=1)
36 train_setLabel = train_set['HeartDisease'].copy()
37
38 #using LogisticRegression model
39 model = LogisticRegression(max_iter = 1000)
40 model.fit(final_train_set,train_setLabel)
41
42 #creating a data frame for the values taken from database
43 cpt = 0
44 sex = 0
45 recg = 0
46 exag = 0
47 slp = 0
48
49 if tpl[1] == 'TA':
50     cpt = 1
51 elif tpl[1] == 'ATA':
52     cpt = 2
53 elif tpl[1] == 'NAP':
54     cpt = 3
55 elif tpl[1] == 'ASY':
56     cpt = 4
57
58 if tpl[11] == 'M':
59     sex = 1
60 elif tpl[11] == 'F':
61     sex = 0
62
63 if tpl[5]=='Normal':
64     recg = 0
65 elif tpl[5]=='ST':
```

```

55 elif tpl[1] == 'ASY':
56     cpt = 4
57
58 if tpl[11] == 'M':
59     sex = 1
60 elif tpl[11] == 'F':
61     sex = 0
62
63 if tpl[5] == 'Normal':
64     recg = 0
65 elif tpl[5] == 'ST':
66     recg = 1
67 elif tpl[5] == 'LVH':
68     recg = 2
69
70 if tpl[7] == 'N':
71     exag = 0
72 elif tpl[7] == 'Y':
73     exag = 1
74
75 if tpl[9] == 'Flat':
76     slp = 0
77 elif tpl[9] == 'Up':
78     slp = 1
79 elif tpl[9] == 'Down':
80     slp = -1
81
82 #dataframe creation
83 data = {'AGE': [tpl[10]], 'Sex': [sex], 'ChestPainType': [cpt], 'RestingBP': [tpl[2]], 'Cholesterol': [tpl[3]], 'FastingBS': [eval(tpl[4])], 'RestingECG': [recg], 'Exag': [exag], 'Slope': [slp]}
84 dt = pd.DataFrame(data)
85 p = model.predict(dt) #passing patient lab test info in model for prediction
86 return p[0] #returning the predicted result

```

9.2 Classification Model used

Table 9.1 Outputs of tested models

Model name	Accuracy Score
Logistic Regression	0.8478260869565217
Decision Tree Classifier	0.7934782608695652
KNN	0.717391304347826

From the above table we can observe that Logistic Regression model is performing better than the other models.

Table 9.2 Classification report of Logistic Regression Model

Value	Precision	Recall	F1-score	Support
0	0.78	0.84	0.81	38
1	0.88	0.83	0.86	54

9.3 Output of Prediction Module:

The screenshot shows a web application window titled "REPORT GENERATION". It has a blue header bar with the title and a "Go back" button in the top right corner. The main content area is light gray and contains two input fields at the top. The first field is labeled "Enter patient ID :" and contains the text "ID0005". Below it is a blue "Report" button. The second field is labeled "Enter name of the patient to find patient ID ::" and is empty. Below it is a blue "FIND" button. In the center, there is an "OUTPUT ::" label followed by a text box containing the message "Patient with patient ID 'ID0005' does not have Heart Disease". The text box has a scroll bar on the right. At the bottom of the window is a Windows taskbar showing various icons and the system clock indicating 31°C Haze, ENG, and 23:05.

REPORT GENERATION

Go back

Enter patient ID : ID0005

Report

Enter name of the patient to find patient ID ::

FIND

OUTPUT :: Patient with patient ID 'ID0005' does not have Heart Disease

This screenshot shows the same "REPORT GENERATION" application window. The "Enter patient ID :" field now contains "ID0001". The "Report" button is still present. The "OUTPUT ::" text box now displays the message "Patient with patient ID 'ID0001' has Heart Disease." The "FIND" button and the rest of the interface remain the same as in the previous screenshot.

REPORT GENERATION

Go back

Enter patient ID : ID0001

Report

Enter name of the patient to find patient ID ::

FIND

OUTPUT :: Patient with patient ID 'ID0001' has Heart Disease.

Chapter 10

TESTING

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance.

Black Box Testing

It is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

White Box Testing

It is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

10.1 Test Case Design

A test case is designed with intent of finding errors or bugs in the software. A test case specifies the inputs, the expected output, and the actual output. If the output given by the software matches the output expected then the software passes that test case.

10.1.1 Boundary Value Analysis

Faults generally occur at the boundary values hence it is essential to test the software at nominal as well as boundary values. In this project, the heart disease is predicted on the basis of ten attributes.

(Age + Sex + ChestPainType + RestingBP + Cholesterol + FastingBS + RestingECG + MaxHR + ExerciseAngina + Oldpeak + ST_Slope)

Table 10.1 Test Case

Test Case	Age	Sex	ChestPain Type	Resting BP	Cholesterol	Fasting BS	Resting ECG	MaxHR	Exercise Angina	Oldpeak	ST_Slope	Predicted Output	Expected Output
1.	53	0	2	113	468	0	0	127	0	0	1	0	0
2.	50	0	2	110	202	0	0	145	0	0	1	0	0
3.	46	1	1	140	272	1	0	175	0	2	0	1	1
4.	37	0	2	120	260	0	0	130	0	0	1	0	0
5.	63	1	4	110	252	0	1	140	1	2	0	1	1
6.	48	1	4	140	208	0	0	159	1	1.5	1	1	1
7.	68	1	1	139	181	1	1	135	0	0.2	1	0	0
8.	62	1	2	120	254	0	2	93	1	0	0	1	1
9.	69	1	3	140	255	1	1	118	0	2.5	-1	1	1
10.	65	1	4	155	255	0	0	154	0	1	1	0	0
11.	53	1	4	140	243	0	0	155	0	0	1	0	0
12.	39	1	2	120	241	0	1	146	0	2	1	0	0
13.	53	1	2	140	320	0	0	162	0	0	1	0	0
14.	46	1	4	180	280	0	1	120	0	0	1	0	0
15.	63	1	4	110	252	0	1	140	1	2	0	1	1
16.	34	1	1	118	182	0	2	174	0	0	1	0	0
17.	73	0	3	160	255	0	1	121	0	0	1	0	1
18.	53	1	2	130	255	0	1	120	0	0.7	-1	1	0
19.	61	1	4	146	241	0	0	148	1	3	-1	1	1
20.	58	0	4	130	197	0	0	131	0	0.6	0	0	0

10.2 Flow Graph

Flow Graph is defined as a function in a program that can be represented as a control flow graph and the nodes in the flow graph are defined as program statements while the directed edges are the flow of control.

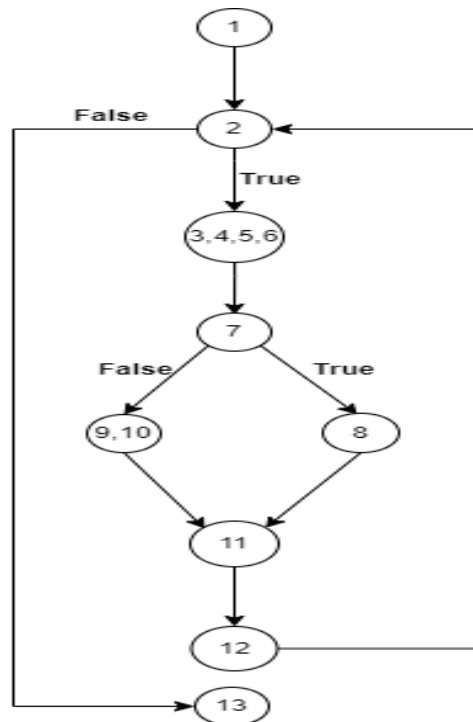


Figure 10.1 : Pseudo code Flow graph

10.3 Basis Path Set

Basis path testing is a technique of selecting the paths in the flow graph, that provide a bases set of execution paths through the program or module.

Basis Path Set:-

1. $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 11 \rightarrow 12 \rightarrow 2 \rightarrow 13$
2. $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 2 \rightarrow 13$
3. $1 \rightarrow 2 \rightarrow 13$

10.4 Cyclomatic Complexity

Cyclomatic complexity of a code section is the quantitative measure of the number of linearly independent paths in it. It is a software metric used to indicate the complexity of a program. It is computed using the Control Flow Graph of the program.

Calculating cyclomatic complexity ($V(G)$) of the pseudocode:

Method 1. $V(G)$ = Total number of regions (closed/open) in the flow graph

We have,

Close regions = 2

Open region = 1

Then,

$$V(G) = 2 + 1 = 3$$

Method 2. $V(G) = \text{Number of predicate nodes} + 1$

We have,

Number of predicate nodes = 2

Then,

$$V(G) = 2 + 1 = 3$$

Method 3. $V(G) = e - n + 2p$

Where,

e = the number of edges in the flow graph

n = the number of nodes in the flow graph

p = the number of connected components

We have,

$$e = 10, n = 9, p = 1$$

Then,

$$V(G) = 10 - 9 + 2 \cdot 1 = 3$$

Chapter 11

REFERENCES

1. Software Engineering- A Practitioner's Approach by Roger S. Pressman: 8th Edition Mc Graw Hill 2005
2. Software Engineering by K.K. Aggarwal & Yogesh Singh
3. S. Kumar, "Predicting and diagnosing of heart disease using machine learning algorithms". International Journal of Engineering and Computer Science, vol. 6, no. 6, pp. 2319-7242, 2017.

View at: Publisher Site | Google Scholar

4. Dataset – <http://www.kaggle.com>
5. GeeksforGeeks – <http://www.geeksforgeeks.org>