BLOOD PRESSURE MONITORING SYSTEM

A

Project Report

Submitted In Partial Fulfillment of the Requirements

Bachelor of Technology

Under Guidance Of

SOUVIK GANGULY

MICROSOFT & HPE CERTIFIED TECHNICAL TRAINER

Submitted By-

Aniket Bagani

ACKNOWLEDGEMENT

1. Title of the project: Blood Pressure Monitoring System
2. Name of the Guide: Mr. Souvik Ganguly
3. Educational Qualification of the Guide:

PhD	MTech	MCA	MSc	MBA	BTech
☐	☐	☐	☐	☐	☐

4. Working/ Teaching experience of the Guide: 4 years

5. Software used in this project
   a. Python 3.7
   b. Google colab
   c. Google doc

For office use only	Signature of the Guide
Approved	…………………………
Not Approved	

Self Certificate

This is to certify that the dissertation/project proposal entitled **“Blood Pressure Monitoring System” is done by Aniket Bagani** is an authentic work carried out for the partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** under the guidance of **Mr. Souvik Ganguly**. The matter embodied in this project work has not been submitted earlier for the award of any degree to the best of my knowledge and belief.

Name of the Student
Aniket Bagani

Certificate by Guide

This is to certify that this project entitled **“Blood Pressure Monitoring System” was** submitted in partial fulfillment of the degree of **Bachelor of Technology (B.Tech)** by **Maulana Abul Kalam Azad University of Technology** through **College of Engineering and Management, Kolaghat done by Aniket Bagani**

Is an authentic work carried out under my guidance & best of our knowledge and belief.

Signature of Student	Signature of the Guide
Aniket Bagani	
Date: 10.06.2022	Date:

Certificate of Approval

This is to certify that this documentation of **Summer Vacation Training Program 2021**, entitled **"Blood Pressure Monitoring System"** is a record of bona-fide work, carried out by **Aniket Bagani**under my supervision and guidance.

In my opinion, the report in its present form is in fulfillment of all the requirements, as specified by the **Maulana Abul Kalam Azad University of Technology.** In fact, it has attained the standard, necessary for submission. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report for **Bachelor of Technology**.

……………………………………………………………..

Mr. Souvik Ganguly

Microsoft & HP Certified Technical

TABLE OF CONTENTS

1. ABSTRACT

Blood pressure (BP) is currently measured using sphygmomanometers, and it is a crucial biomarker of a person's heart health. Hence, regular monitoring of blood pressure is important for early diagnosis and treatment. On the other hand, conventional blood pressure measurement devices discomfort patients, since the blood flow is cut off with the pressure exerted by the cuff while measuring systolic blood pressure. Nowadays, researchers are using different signals such as Electrocardiogram (ECG) and Photoplethysmography (PPG) to extract useful information like pulse arrival time (PAT) and pulse transit time (PTT) in order to estimate blood pressure without using a cuff. Two signals can be used simultaneously, but this method requires two sensors, which makes it expensive and impractical. To overcome this, only PPG-based cuffless and continuous monitoring of blood pressure has been proposed in several studies. In this paper, in order to estimate systolic and diastolic blood pressure values, three different machine learning algorithms, i.e. Linear Regression (LR), Support Vector Regression (SVR), and Artificial Neural Networks (ANNs), were implemented using PPG signals and some other features such as body mass index (BMI), age, height and weight obtained from the patient. A new, short-recorded photoplethysmogram dataset was used for this purpose, and the results are compared in terms of mean absolute error.

2. INTRODUCTION

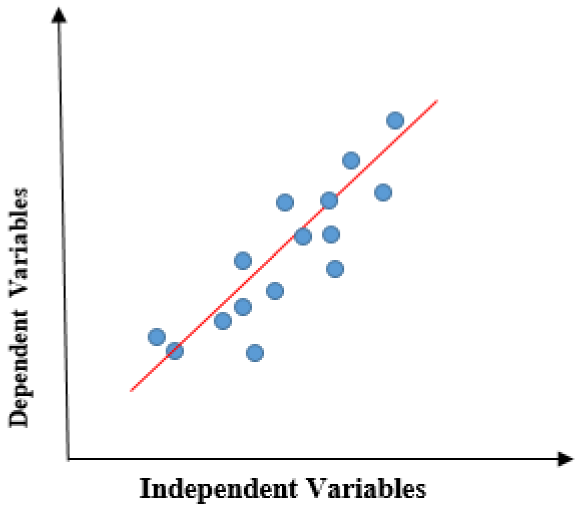
Blood pressure diseases have increasingly been identified as among the main factors threatening human health. How to accurately and conveniently measure blood pressure is the key to the implementation of effective prevention and control measures for blood pressure diseases. Traditional blood pressure measurement methods exhibit many inherent disadvantages, for example, the time needed for each measurement is difficult to determine, continuous measurement causes discomfort, and the measurement process is relatively cumbersome. Wearable devices that enable continuous measurement of blood pressure provide new opportunities and hopes. Although machine learning methods for blood pressure prediction have been studied, the accuracy of the results does not satisfy the needs of practical applications.

3. ALGORITHMS

For our project, we focused on only one algorithm: Linear Regression Algorithm.

3.1 Linear Regression:

Linear regression is a quiet and simple statistical regression method used for predictive analysis and shows the relationship between the continuous variables. Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), consequently called linear regression. If there is a single input variable (x), such linear regression is called simple linear regression. And if there is more than one input variable, such linear regression is called multiple linear regression. The linear regression model gives a sloped straight line describing the relationship within the variables.



The above graph presents the linear relationship between the dependent variable and independent variables. When the value of x (independent variable) increases, the value of y (dependent variable) is likewise increasing. The red line is referred to as the best fit straight line. Based on the given data points, we try to plot a line that models the points the best.

To calculate the best-fit line linear regression uses a traditional slope-intercept form.

$$y = mx+b \Longrightarrow y = a_0+a_1x$$

y= Dependent Variable. x= Independent Variable. a0= intercept of the line.

a1 = Linear regression coefficient.

4. PROBLEM STATEMENT

Efficient blood pressure prediction method based on the Linear Regression algorithm to solve the key gap between the need for continuous measurement for prophylaxis and the lack of an effective method for continuous measurement. The results of the algorithm were compared with those obtained from one classical machine learning algorithm, i.e., back propagation neural network (BP), with respect to six evaluation indexes (accuracy, pass rate, mean absolute percentage error (MAPE), mean absolute error (MAE), R-squared coefficient of determination (R2) and Spearman’s rank correlation coefficient). The experimental results showed that the Linear Regression model can accurately and effectively predict blood pressure.

5. BACKGROUND DATASET

- bdp is a data set that provides 150 records of Data.
- There are 3 columns: Pulse, Systolic Pressure, and Diastolic Pressure.
- The values are Real Values.
- The Dataset has no null values.
- We’ve applied Linear Regression Algorithm to the Dataset.

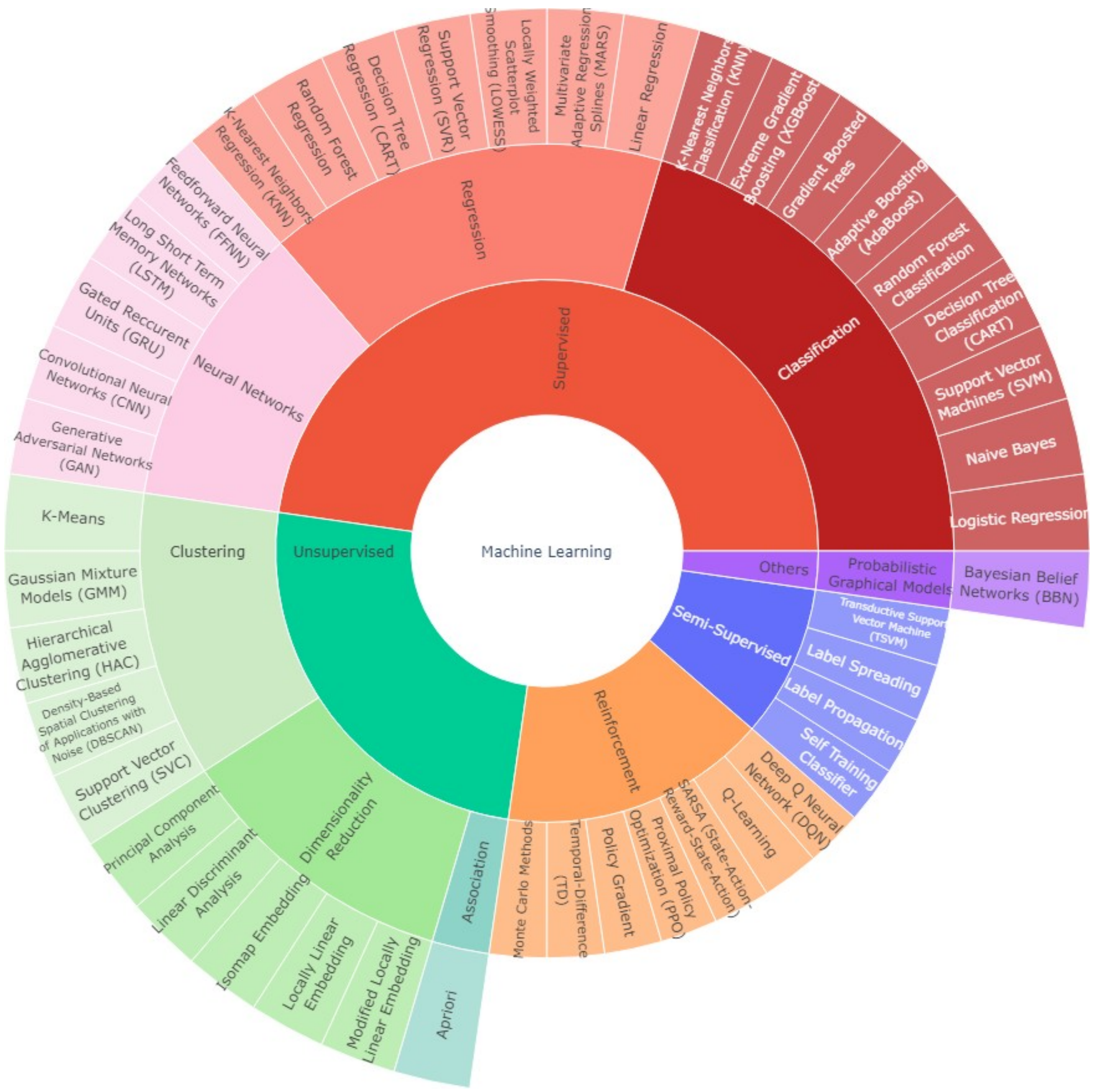
6. WHAT IS ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING?

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly.

The word Artificial Intelligence comprises of two words “Artificial” and “Intelligence”. Artificial refers to something which is made by humans or a non-natural thing and Intelligence means the ability to understand or think. There is a misconception that Artificial Intelligence is a system, but it is not a system. AI is implemented in the system. There can be so many definitions of AI, one definition can be “It is the study of how to train the computers so that computers can do things which at present humans can do better.” Therefore It is an intelligence that we want to add all the capabilities to a machine that human contains.


Machine Learning
Supervised
Unsupervised
Semi-Supervised
Reinforcement
Others
Regression
Classification
Neural Networks
Clustering
Dimensionality Reduction
Association
Probabilistic Graphical Models
K-Nearest Neighbors Regression (KNN)
Random Forest Regression
Decision Tree Regression (CART)
Support Vector Regression (SVR)
Locally Weighted Scatterplot Smoothing (LOWESS)
Multivariate Adaptive Regression Splines (MARS)
Linear Regression
K-Nearest Neighbors Classification (KNN)
Extreme Gradient Boosting (XGBoost)
Gradient Boosted Trees
Adaptive Boosting (AdaBoost)
Random Forest Classification
Decision Tree Classification (CART)
Support Vector Machines (SVM)
Naive Bayes
Logistic Regression
Bayesian Belief Networks (BBN)
Transductive Support Vector Machine (TSVM)
Label Spreading
Label Propagation
Self Training Classifier
Deep Q Neural Network (DQN)
Q-Learning
SARSA (State-Action-Reward-State-Action)
Proximal Policy Optimization (PPO)
Policy Gradient
Temporal-Difference (TD)
Monte Carlo Methods
Apriori
Modified Locally Linear Embedding
Locally Linear Embedding
Isomap Embedding
Linear Discriminant Analysis
Principal Component Analysis
Support Vector Clustering (SVC)
Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
Hierarchical Agglomerative Clustering (HAC)
Gaussian Mixture Models (GMM)
K-Means
Generative Adversarial Networks (GAN)
Convolutional Neural Networks (CNN)
Gated Reccurent Units (GRU)
Long Short Term Memory Networks (LSTM)
Feedforward Neural Networks (FFNN)

7. SYSTEM ANALYSIS

7.1 IDENTIFICATION OF NEED:

System analysis is a process of gathering and interpreting facts, diagnosing problems, and the information to recommend improvements to the system. It is a problem-solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The System is viewed as a whole and the input to the system is identified. The outputs from the organization are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and Decisional variables, analyzing and synthesizing the various factors, and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews, questionnaires, etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is a loop that ends as soon as the user is satisfied with the proposal.

7.2 FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work

Effort and time spent on it: A feasibility study lets the developer foresee the future of the project and its usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet its user needs and effective use of resources. Thus when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provides the feasibility of the project that is being designed and lists various area that was considered very carefully during the feasibility study of this project such as Technical, Economic, and operational feasibilities.

Technical Feasibility:

This project is technically feasible as all it has got to do to extract tweets, is to get the proper credentials from the Developer’s console provided by Twitter. After the credentials are obtained, i.e. the Access Token Key, Access Token Secret Key, Consumer Key, and Consumer Secret key, Twitter gives us access to its tweets. Hence we get a sufficiently large dataset to conduct sentiment analysis. Also the range of tweets obtained is limited to 300 tweets per page, which ensures that the results do not go out of bounds. Thus, this is technically feasible.

Economic Feasibility:

This project work is economically feasible as it does not take into account any additional costs. Whatever data is extracted, it is done without any charges. Twitter provides free use of this data that is non-encrypted and publicly available for analysis purposes. Hence, this work is economically feasible as well.

Operational Feasibility:

This is operationally feasible as well. As already mentioned, it takes in 300 tweets per page as that is the limit set by Twitter. Therefore it is operationally feasible as well. The system won’t hang when getting the results.

7.3 WORKFLOW

◦ First we have uploaded the bdp dataset and accessed the information of the dataset.

◦ We apply the Regression Algorithm to the dataset and split the dataset into training and testing parts.

◦ Now we plot the pulse vs diastolic Pressure on training and testing data.

◦ Then we fit the model and predicted the values of systolic and diastolic pressure.

◦ Finally, we measure the mean squared error and then again predict the values.

7.4 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements

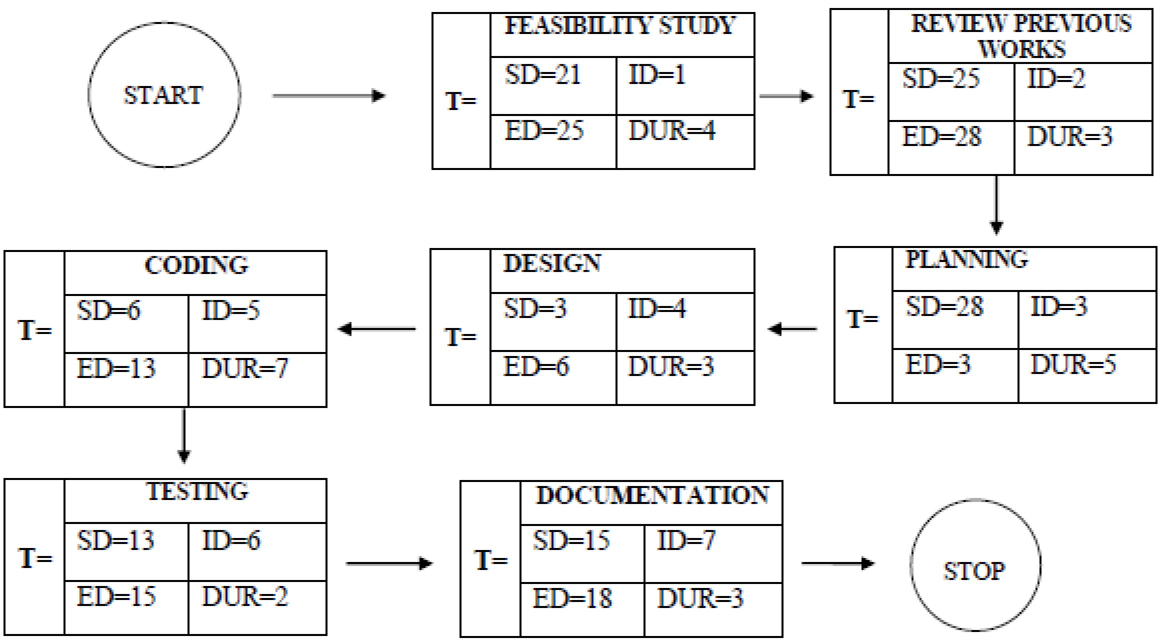
A standard computer with at least i3 processor Standard computer with 2GB of RAM Standard computer with 100GB of free space

Software Requirements

- python 3.9
- Google Colab
- Google doc

8. SYSTEM DESIGN

PART CHART



PERT CHART DIAGRAM

SEQUENCE DIAGRAM

Sequence diagrams can be useful reference diagrams for businesses and other organizations. Try drawing a sequence diagram to

- Represent the details of a UML use case.

• Model the logic of a sophisticated procedure, function, or operation.

• See how tasks are moved between objects or components of a process.

• Plan and understand the detailed functionality of an existing or future scenario.

Popular Sequence Diagram Uses

Usage Scenario –

A usage scenario is a diagram of how your system could potentially be used. It’s a great way to make

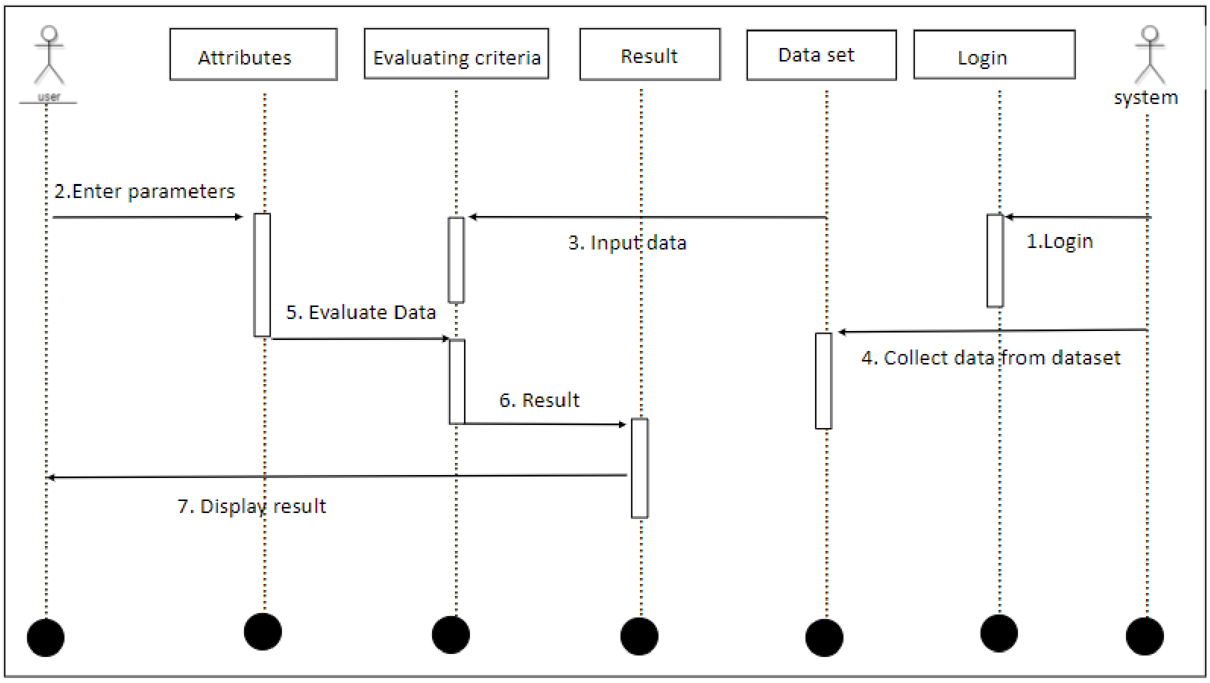
sure that you have worked through the logic of every usage scenario for the system.

Method Logic –

Just as you might use a UML sequence diagram to explore the logic of a use case, you can use it to Usage Scenario - A usage scenario is a diagram of how your system could potentially be used. It's a great explore the logic of any function, procedure, or complex process.

Service Logic –

If you consider a service to be a high-level method used by different clients, a sequence diagram is an ideal way to map that out.



ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

Purpose of Activity Diagrams

The basic purpose of activity diagrams is similar to the other four diagrams. It captures the dynamic behavior of the system. The other four diagrams are used to show the message flow from one object to another but the activity diagram is used to show the message flow from one activity to another.

Where to Use Activity Diagrams?

The basic usage of the activity diagram is similar to the other four UML diagrams. The specific usage is to model the control flow from one activity to another. This control flow does not include messages.

An activity diagram is suitable for modeling the activity flow of the system. An application can have multiple systems. The activity diagram also captures these systems and describes the flow from one system to another. This specific usage is not available in other diagrams. These systems can be databases, external queues, or any other system.

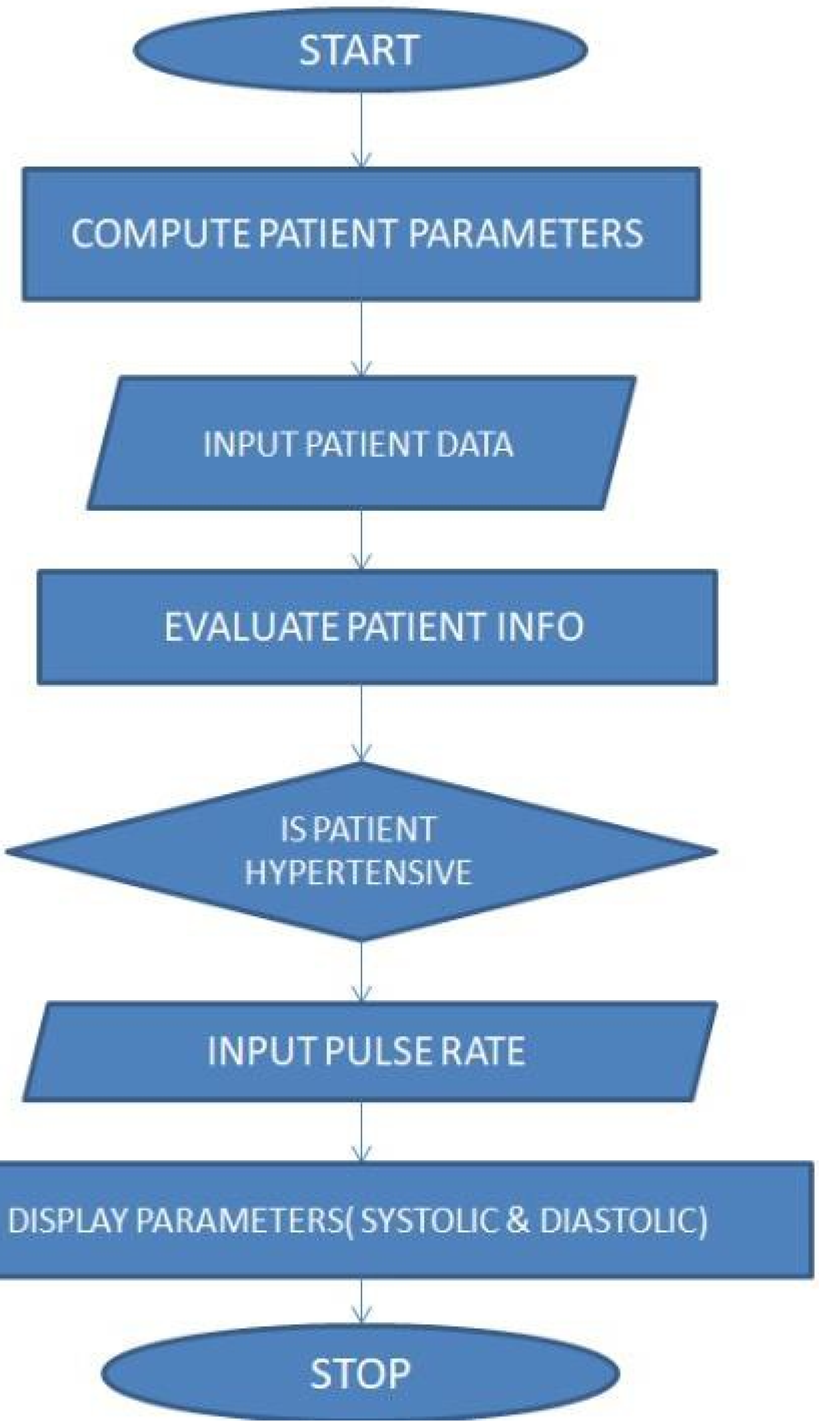


Fig: Activity Diagram

USE CASE DIAGRAM

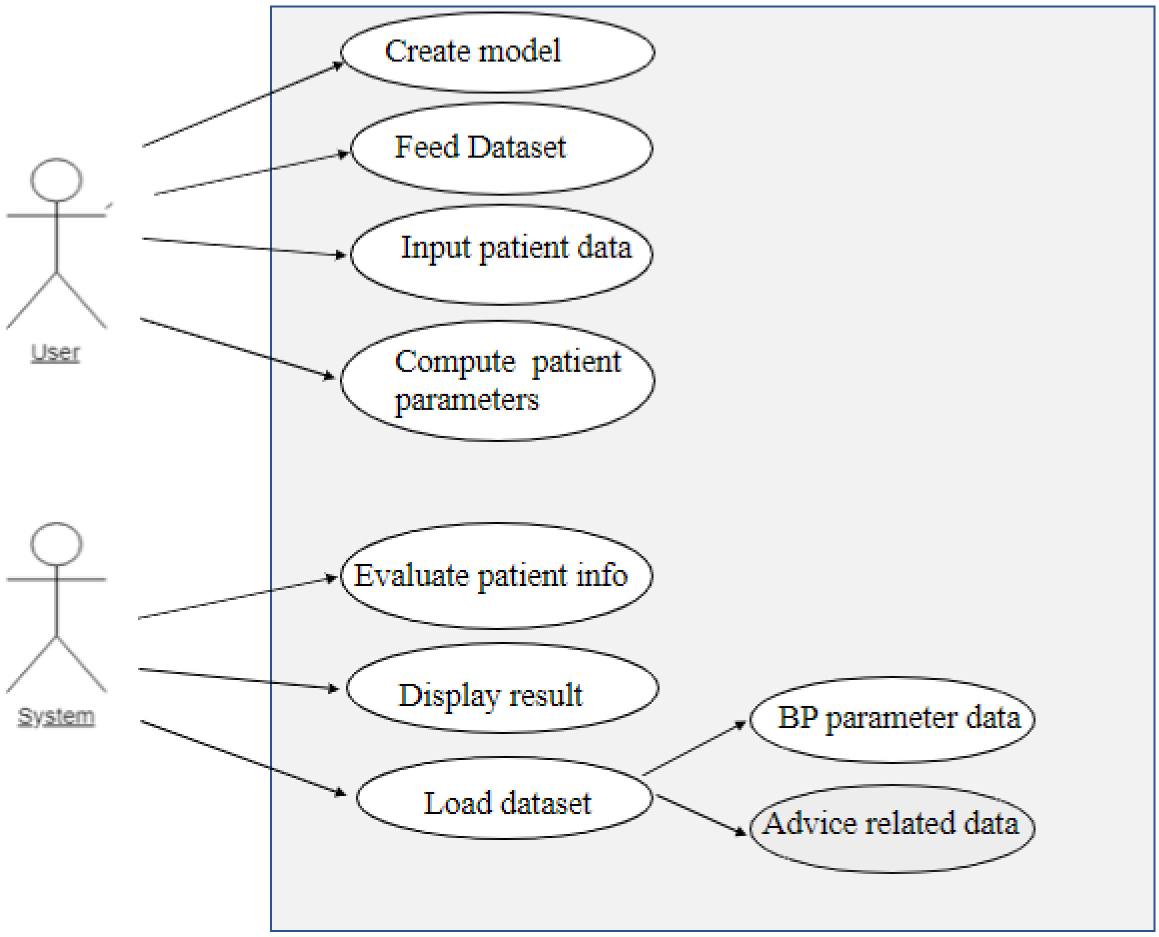
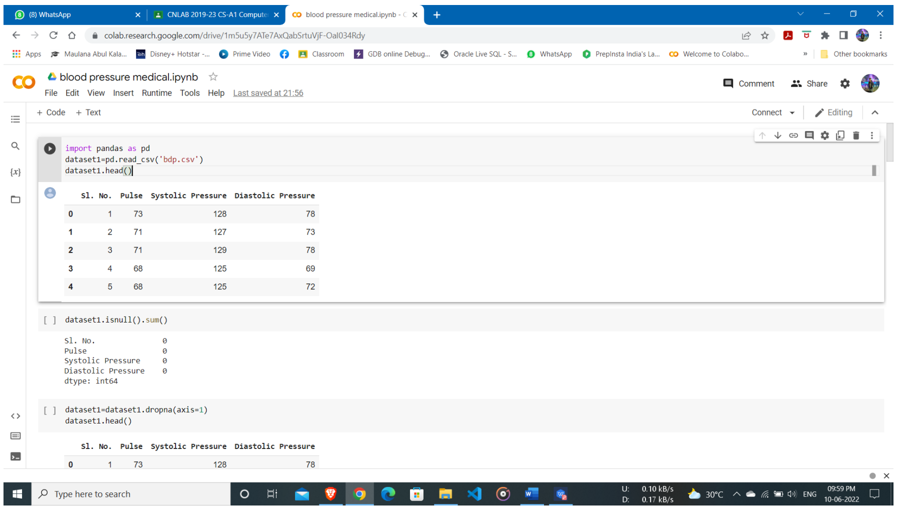


Fig: Use Case Diagram

9. USER INTERFACE DESIGN

1. The design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centered design).
2. Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. Graphic design and typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the interface. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Snapshots:



```
import pandas as pd
dataset1=pd.read_csv('bdp.csv')
dataset1.head()
```

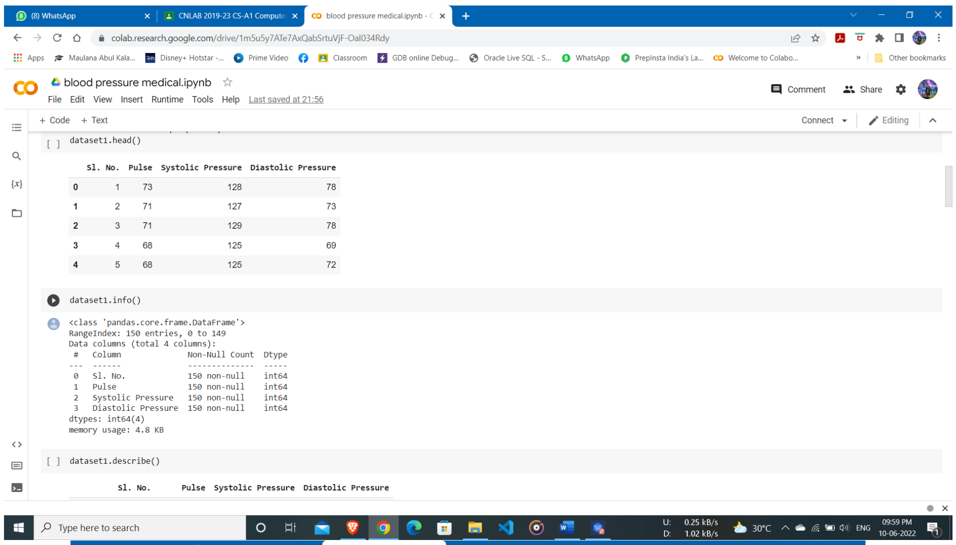
	Sl. No.	Pulse	Systolic Pressure	Diastolic Pressure
0	1	73	128	78
1	2	71	127	73
2	3	71	129	78
3	4	68	125	69
4	5	68	125	72

```
dataset1.isnull().sum()
```

```
Sl. No.               0
Pulse                 0
Systolic Pressure     0
Diastolic Pressure    0
dtype: int64
```

```
dataset1=dataset1.dropna(axis=1)
dataset1.head()
```

	Sl. No.	Pulse	Systolic Pressure	Diastolic Pressure
0	1	73	128	78



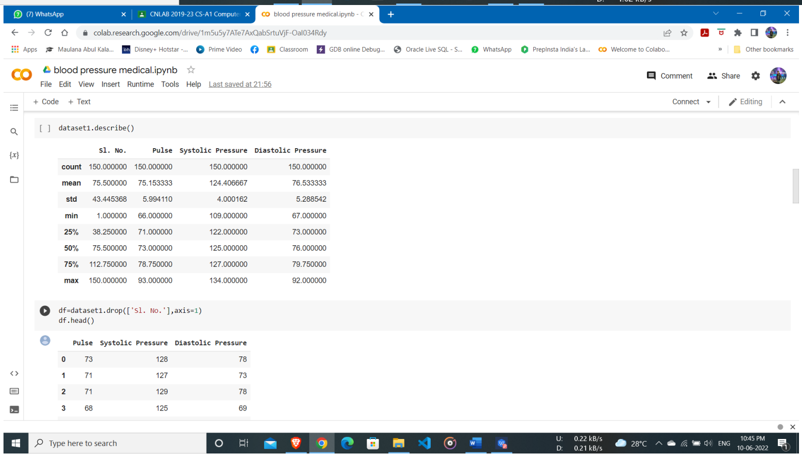
```
dataset1.head()
```

	Sl. No.	Pulse	Systolic Pressure	Diastolic Pressure
0	1	73	128	78
1	2	71	127	73
2	3	71	129	78
3	4	68	125	69
4	5	68	125	72

```
dataset1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Sl. No.             150 non-null    int64
 1   Pulse               150 non-null    int64
 2   Systolic Pressure   150 non-null    int64
 3   Diastolic Pressure  150 non-null    int64
dtypes: int64(4)
memory usage: 4.8 KB
```

```
dataset1.describe()
```

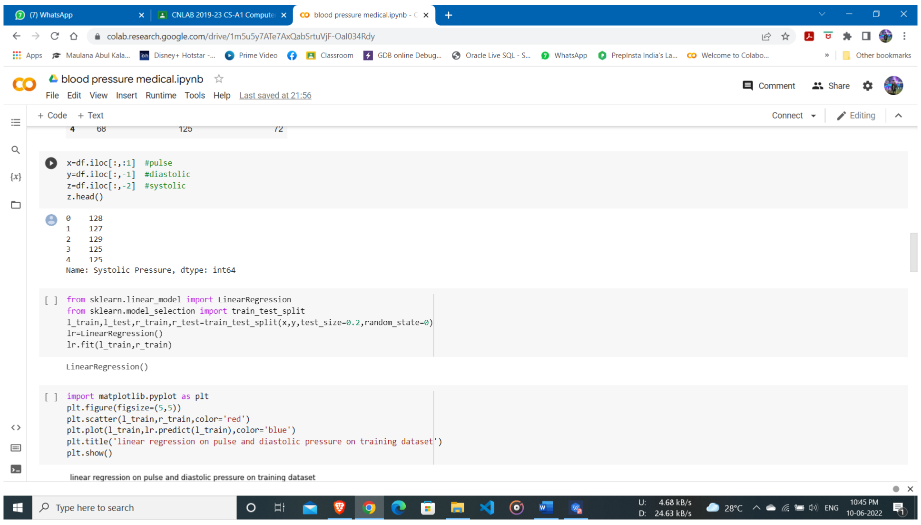


```
dataset1.describe()
```

	Sl. No.	Pulse	Systolic Pressure	Diastolic Pressure
count	150.000000	150.000000	150.000000	150.000000
mean	75.500000	75.153333	124.406667	76.533333
std	43.445368	5.994110	4.000162	5.288542
min	1.000000	66.000000	109.000000	67.000000
25%	38.250000	71.000000	122.000000	73.000000
50%	75.500000	73.000000	125.000000	76.000000
75%	112.750000	78.750000	127.000000	79.750000
max	150.000000	93.000000	134.000000	92.000000

```
df=dataset1.drop(['Sl. No.'],axis=1)
df.head()
```

	Pulse	Systolic Pressure	Diastolic Pressure
0	73	128	78
1	71	127	73
2	71	129	78
3	68	125	69



4	68	125	72

```
x=df.iloc[:,:1]  #pulse
y=df.iloc[:,-1]  #diastolic
z=df.iloc[:,-2]  #systolic
z.head()
```

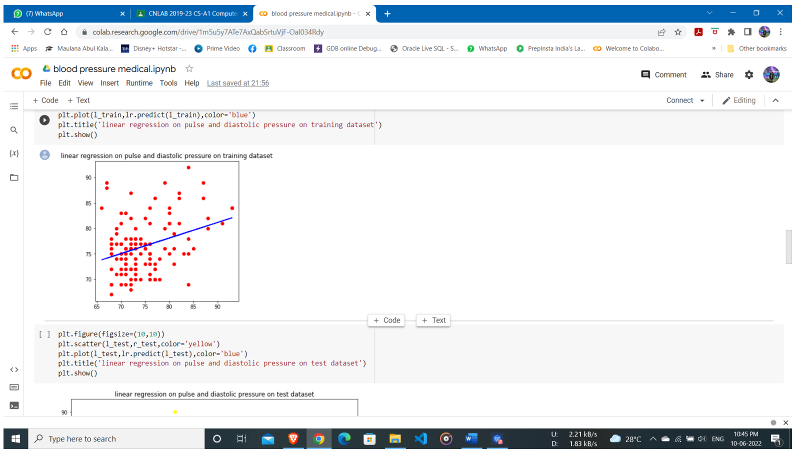
```
0    128
1    127
2    129
3    125
4    125
Name: Systolic Pressure, dtype: int64
```

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
l_train,l_test,r_train,r_test=train_test_split(x,y,test_size=0.2,random_state=0)
lr=LinearRegression()
lr.fit(l_train,r_train)
```

```
LinearRegression()
```

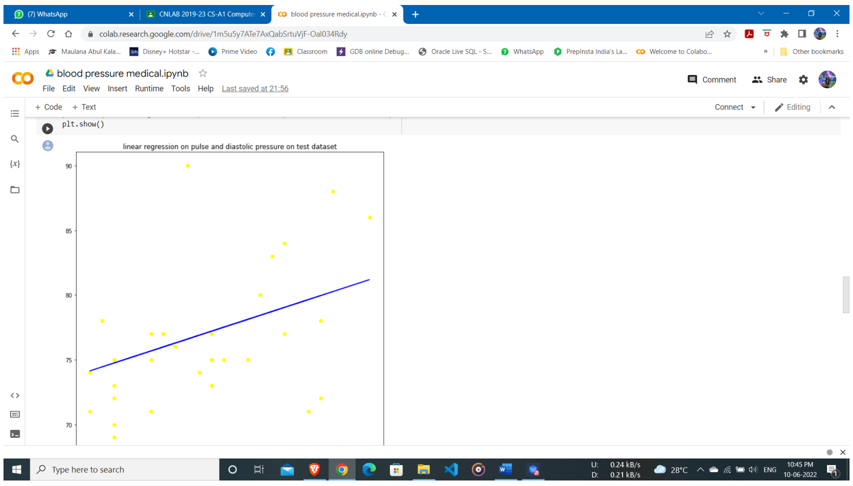
```
import matplotlib.pyplot as plt
plt.figure(figsize=(5,5))
plt.scatter(l_train,r_train,color='red')
plt.plot(l_train,lr.predict(l_train),color='blue')
plt.title('linear regression on pulse and diastolic pressure on training dataset')
plt.show()
```

linear regression on pulse and diastolic pressure on training dataset

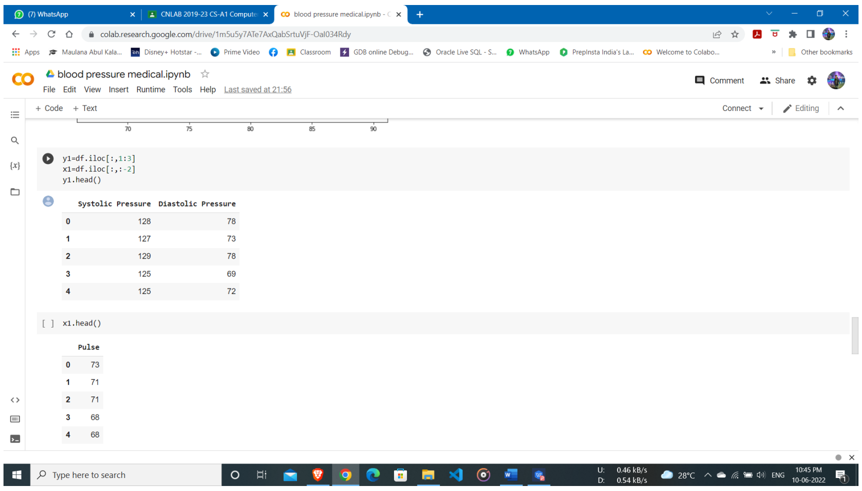


```
plt.plot(l_train,lr.predict(l_train),color='blue')
plt.title('linear regression on pulse and diastolic pressure on training dataset')
plt.show()
```

```
[ ] plt.figure(figsize=(10,10))
    plt.scatter(l_test,r_test,color='yellow')
    plt.plot(l_test,lr.predict(l_test),color='blue')
    plt.title('linear regression on pulse and diastolic pressure on test dataset')
    plt.show()
```



```
plt.show()
```

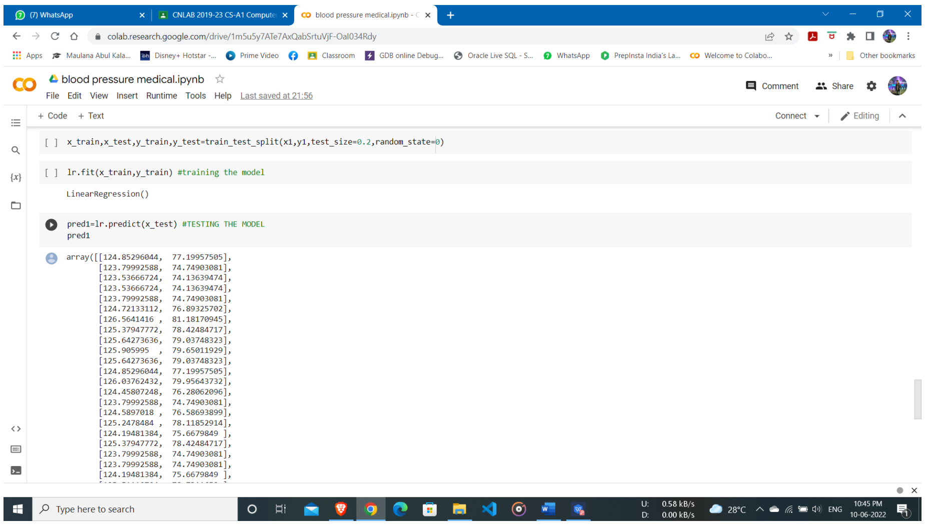


```
y1=df.iloc[:,1:3]
x1=df.iloc[:,:-2]
y1.head()
```

	Systolic Pressure	Diastolic Pressure
0	128	78
1	127	73
2	129	78
3	125	69
4	125	72

```
[ ] x1.head()
```

	Pulse
0	73
1	71
2	71
3	68
4	68



```
[ ] x_train,x_test,y_train,y_test=train_test_split(x1,y1,test_size=0.2,random_state=0)

[ ] lr.fit(x_train,y_train) #training the model

LinearRegression()

pred1=lr.predict(x_test) #TESTING THE MODEL
pred1

array([[124.85296044,  77.19957505],
       [123.79992588,  74.74903081],
       [123.53666724,  74.13639474],
       [123.53666724,  74.13639474],
       [123.79992588,  74.74903081],
       [124.72133112,  76.89325702],
       [126.5641416 ,  81.18170945],
       [125.37947772,  78.42484717],
       [125.64273636,  79.03748323],
       [125.905995  ,  79.65011929],
       [125.64273636,  79.03748323],
       [124.85296044,  77.19957505],
       [126.03762432,  79.95643732],
       [124.45807248,  76.28062096],
       [123.79992588,  74.74903081],
       [124.5897018 ,  76.58693899],
       [125.2478484 ,  78.11852914],
       [124.19481384,  75.6679849 ],
       [125.37947772,  78.42484717],
       [123.79992588,  74.74903081],
       [123.79992588,  74.74903081],
       [124.19481384,  75.6679849 ],
```

10. IMPLEMENTATION AND TESTING

A software system test plan is a document that describes the objectives, scope, approach, and focus of the software testing effort. The process of preparing a test plan is a usual way to think about the effort needed to validate the acceptability of a software product. The complete document will help people outside the test group understand the "WHY" and "HOW" product validation. It should be thorough enough to be useful but not so thorough that no one outside the test group will read it.

Introduction

Testing is the process of running a system with the intention of finding errors. Testing enhances the integrity of a system by detecting deviations in design and errors in the system. Testing aims at detecting error-prone areas. This helps in the prevention of errors in a system. Testing also adds value to the product by conforming to the user requirements. The main purpose of testing is to detect errors and error-prone areas in a system. Testing must be well planned. A partially tested system is to detect errors and error-prone areas in a system. Testing must be well planned. A partially tested system is as bad as an untested system. And the price of an untested and under-tested system is high.

Objectives of Testing

The objective of our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

Process Overview

The following represents the overall flow of the testing process:
• Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
• Identify which particular test(s) will be used to test each module.
• Review the test data and test cases to ensure that the unit has been thoroughly verified and that the
test data and test cases are adequate to verify proper operation of the unit.

Test Cases

A test case is a document that describes an input, action, or event and expected response, to determine if a feature of an application is working correctly. A test case should contain particular such as test case identifier, test condition, and input data.
Requirement expected results. The process of developing test cases can help find problems in the requirements or design of an application since it requires completely thinking through the operations of the application.

Testing Steps

- ### Unit Testing

Unit testing focuses efforts on the smallest unit of software design. This is known as module testing. The modules are tested separately. The test is carried out during programming stage itself. In this step, each module is found to be working satisfactory as regards to the expected output from the module.

- ### Integration Testing

Data can be lost across an interface. One module can have an adverse effect on another, sub functions, when combined, may not be linked in the desired manner in major functions. Integration testing is a systematic approach for constructing the program structure, while at the same time conducting tests to uncover errors associated with the interface.

Validation

At the culmination of the integration testing, the Software is completely assembled as a package.
Interfacing errors have been uncovered and corrected and a final series of software tests begin in validation testing. Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After the validation test has been conducted, one of the three possible conditions exists a)The function or performance characteristics conform to specification and are accepted.
b) A deviation from the specification is uncovered and a deficiency list is created.
c) Proposed system under consideration has been tested by using a validation test and found to be working satisfactorily.

Tested By:	Aniket Bagani	
Test Type	Unit Testing	
Test CaseNumber	1	
TestCaseName	Blood Pressure Monitoring system	
Test Case Description	The user should enter the pulse rate. The system will then check which genre it belongs to. Accordingly it will give the output and the user can give it a check whether it's accurate or not.	
Item(s) to be tested		
1	Verification of pulse rate provided by the system.	
Specifications		
Input		Expected Output/Result
Enter the pulse rate Search for the genre of the input pulse rate		Output contain systolic and diastolic pressure values or rate Accurate systolic and diastolic rate will be displayed in the output

White Box Testing

In the white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. To ensure this happens, we will be applying Branch Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.

Black box testing

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity.

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format. The output format on the screen is found to be correct. The format was designed in the system design time according to the user needs. For the hard copy also; the output comes as per the specified requirements by the user. Hence output testing did not result in any correction for the system.

User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required. This is done in regard to the following point:

[1] Input Screen Design.

[2] Output Screen Design.

[3] Format of reports and other outputs

Integration Testing

Software testing is always used in association with verification and validation. In the testing phase of this project, our aim is to find the answer to the following two questions.

• Whether the software matches with the specification (i.e.process base) to verify the product.
• Whether this software in one client what wants (i.e. product base) to validate the product.
• Unit testing and integration testing has been carried out to find the answer to the above questions. In unit testing, each individual module was tested to find any unexpected behavior that exists. Later all the module was integrated and a flat file was generated.

Functional Testing

These are the points concerned during the stress test:

• Nominal input: character is in putted in the place of digits and the system has to flash the message "Dataerror"
• Boundary value analysis: exhaustive test cases have designed to create an output report that produces the maximum (and minimum) allowable number of table entries.

11. SYSTEM SECURITY MEASURES

Database Security

System security measure is meant to be provided to make your system reliable and secured from unauthorized user may create threats to the system. So you should follow some security measures. We have used security levels in database level at system level.

System Security

If we talk about the system security in our proposed system we have implemented with the help of maintain the session throughout the system’s use. Once a user has logged out than he/she will not be able to perform any task before signing back again.
A high level of authentic login is given to the system so this is a very tedious task to enter without authorization and authentication.

12 COST ESTIMATION

The Constructive Cost Model (COCOMO) is a procedural software cost estimation model developed by Barry W. Boehm. Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases.

Types of COCOMO:

1. Basic COCOMO:

Basic COCOMO computes software development effort (and cost) as a function of program size. Program size is expressed in estimated thousands of source lines of code (SLOC, KLOC).
COCOMO applies to three classes of software projects:

• Organic projects - small teams with good experience working with less than rigid requirements.
• Semi-detached projects - medium teams with mixed experience working with a mix of rigid and less than rigid requirements.
• Embedded projects - developed within a set of tight constraints. It is also a combination of organic and semi-detached projects. (Hardware, software, operational, ...)

The basic COCOMO equations take the form: Effort Applied (E) = a*(KLOC)b[man-months]
Development Time (D) = c*(Effort Applied)d [months]
People required (P) = Effort Applied / Development Time [count] Where,
KLOC is the estimated number of delivered lines (expressed in thousands) of code for the project. The coefficients a, b, c, and d are given in the following table:

Software Project	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO is good for quick estimate of software costs. However, it does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and so on.

2. Intermediate COCOMO:

Intermediate COCOMO computes software development effort as function of program size and a set of cost drivers that include subjective assessment of product, hardware, personnel and project attributes. This extension considers a set of four cost drivers, each with a number of subsidiary attributes: -

Product attributes:

1) Required software reliability
2) Size of application database
3) Complexity of the product

Hardware attributes:

1) Run-time performance constraints
2) Memory constraints
3) Volatility of the virtual machine environment
4) Required turnabout time

Personnel attributes

1) Analyst capability
2) Software engineering capability
3) Applications experience
4) Virtual machine experience
5) Programming language experience

Project attributes

1. Use of software tools
2. Application of software engineering methods
3. Required development schedule

Each of the 15 attributes receives a rating on a six-point scale that ranges from very low to extra high (in importance or value). An effort multiplier from the table below applies to the rating. The product of all effort multipliers results in an effort adjustment factor (EAF). Typical values for EAF range from 0.9 to 1.4. The Intermediate COCOMO formula now takes the form:

E=a*(KLOC)b(EAF)

Where,

E is the effort applied in person-months, KLOC is the estimated number of thousands of delivered lines of code for the project, and EAF is the factor calculated above. The coefficient a and the exponent b is given in the table:

Software project	a	B
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

The Development time D calculation uses P in the same way as in the Basic COCOMO.
Detailed COCOMO
Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. These Phase Sensitive effort multipliers are each to determine the amount of effort required to complete each phase. In detailed COCOMO, the whole software is divided in different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

In detailed COCOMO, the effort is calculated as function of program size and a set of cost drivers given according to each phase of software life cycle. A Detailed project schedule is never static. The Six phases of detailed COCOMO are: -

- Plan and requirement.
- System design.
- Detailed design.
- Module code and test.
- Integration and test.
- Cost Constructive model

Detailed cost estimation of Movie Recommender System
So by considering all the facts, if we calculate the cost, it will be like;

The OES is having 2.1 Kilo Lines of Code. According to the COCOMO model the comparison is:

Mode	Project Size	Innovatio n	Deadline	Development Environment
Organic	Typically 2-50 KLOC	Little	Not tight	Famili ar and in house
Semi- detached	Typically 50-	Medium	Medium	Medium

	300 KLOC			
Embedded	Typically Over 300 KLOC	Significant	Tight	Complex hardware/Customer interface is required

The project has 2.1 KLOC, so it’s under organic category. Putting the facts in the formulas,
Effort Applied (E) = a*(KLOC)b [person-months]
= 2.4*(2.1)1.05 PM

= 5 PM
Development Time (D) = c*(Effort Applied)d [months]
= 2.5*(5)0.38 M
= 4.6 M

FUTURE SCOPE AND FURTHER ENHANCEMENTS

The blood pressure prediction is going to be very useful and will be an easier way to find out bp for users. Multi-feature joint training and predicting techniques in machine learning can potentially complement and greatly improve the accuracy of traditional blood pressure measurement, resulting in better disease classification and more accurate clinical judgements.

CONCLUSION

Thus the blood pressure prediction got completed successfully. We found that by giving the pulse rate as input, we can get the accurate diastolic and systolic pressure values. Furthermore, this trend has inspired us to use machine learning algorithms to investigate the hidden knowledge in the substantial amount of medical data collected by intelligent devices. These data can assist the medical staff to perform disease diagnosis, especially blood pressure disease prediction and prevention. We can identify and analyze the blood pressure by machine learning mechanisms to establish efficient blood pressure prediction models. It’s going to be helpful in medical terms as it is one of the easier ways to find out blood pressure.

BIBLIOGRAPHY

- Takeo Kanade. volume 47. Birkh¨auser Basel, 1977.
- Lawrence Sirovich and Michael Kirby.
- M. Turk and A. Pentland. Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1):71– 86, Jan 1991.
- Dong chen He and Li Wang. Texture unit, texture spectrum, and texture analysis. IEEE Transactions on Geoscience and Remote Sensing, 28(4):509– 512, Jul 1990.