

PREDICTION OF MULTIPLE DISEASES USING MACHINE LEARNING TECHNIQUES

A MINI PROJECT REPORT

Submitted by

JAFAR SADIQ(111520104052)

CHERUKURU SASIDHAR(111520104024)

HARISH RK(1115201040__)

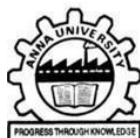
in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



R.M.D. ENGINEERING COLLEGE

(An Autonomous Institution)

KAVARAIPETTAI – 601 206

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2022

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “**PREDICTION OF MULTIPLE DISEASES USING MACHINE LEARNING TECHNIQUES** ” is the bonafide work of “**JAFAR SADIQ(111520104052),CHERUKURU SASIDHAR(111520104024),HARISH RK(1115201040)**” who carried out the project work under my supervision.

SIGNATURE

Dr.P.Ezhumalai M.Tech, F.I.E, Ph.D.,

HEAD OF THE DEPARTMENT

Department of CSE,

R.M.D Engineering College

R.S.M. Nagar,

Kavaraipettai–601206

SIGNATURE



SUPERVISOR

Department of CSE,

R.M.D Engineering College

R.S.M. Nagar,

Kavaraipettai –601206

VIVA – VOCE EXAMINATION

The Viva – Voce Examination of the following
students who have submitted this project work is held
on.....

JAFAR SADIQ	111520104052
CHERUKURU SASIDHAR	111520104024
HARISH RK	1115201040

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

It is our immense pleasure to express our deep sense of gratitude to our chairman **Thiru R. S. MUNIRATHINAM** , our vice chairman **Thiru R.M.KISHORE**, our director **Thiru R. JOTHI NAIDU**, and our Secretary **Thiru YALAMANCHI PRADEEP**, for the facilities and support given by them in the college.

We are extremely thankful to our principal **Dr. N. ANBUCHEZHIAN, B.E., M.S., M.B.A., M.E., Ph.D.**, for having given us an opportunity to Serve the purpose of education.

We take this opportunity to give profound and heartfelt thanks to the Dean-Research **Dr. K. K. THYAGHARAJAN B.E., M.E., Ph.D.**, for their continuous support in successful completion of this project.

We are indebted to **Dr.P.EZHUMALAI B.E., M.Tech., F.I.E., Ph.D.**, Professor and the Head of the Department of Computer Science and Engineering, for his valuable guidance and useful suggestions during the course of the project.

We are thankful to our project supervisor **~~Dr. M.A.BERLIN B.E., M.E., Ph.D.~~**, Associate Professor, Department of Computer Science and Engineering , R.M.D. Engineering College for her helpful guidance and valuable support given to us throughout the project.

ABSTRACT

- Due to machine learning progress in biomedical and healthcare communities, accurate study of medical data benefits early disease recognition, patient care and community services.
- When the quality of medical data is incomplete the exactness of study is reduced. Moreover, different regions exhibit unique appearances of certain regional diseases, which may results in weakening the prediction of disease outbreaks.
- In the proposed system, it provides machine learning algorithms for effective prediction of various disease occurrences in disease-frequent societies. It experiment the altered estimate models over real-life hospital data collected.
- To overcome the difficulty of incomplete data, it use a latent factor model to rebuild the missing data. It experiment on a regional chronic illness of cerebral infarction. Using structured and unstructured data from hospital it use Machine Learning algorithm.
- It predicts probable diseases by mining data sets such as Covid-19, Chronic Kidney disease and heart Disease. To the best of our knowledge in the area of medical big data analytics none of the existing work focused on both data types.
- Compared to several typical estimate algorithms, the calculation exactness of our proposed algorithm reaches 94.8% with a convergence speed which is faster than that of the machine learning disease risk prediction algorithm.

CONTENTS		
CHAPTER NO.	TITLE	PAGE NO.
	TITLE PAGE	i
	BONAFIED CERTIFICATE	ii
	DECLARATION	iii
	ACKNOWLEDGMENT	iv
	ABSTRACT	v
	LIST OF FIGURES	viii
1.	INTRODUCTION	1
	1.1 PROBLEM DEFINITION	2
	1.2 OBJECTIVES	3
	1.2.1 AIM OF THE PROJECT	3
	1.2.2 SCOPE OF THE PROJECT	3
2.	LITERATURE SURVEY	4
3.	EXISTING SYSTEM	8
	3.1 DISADVANTGES OF EXISTING SYSTEM	8
4.	PROPOSED SYSTEM	9
	4.1 ADVANTAGES OF PROPOSED SYSTEM	9
5.	PROPOSED ALGORITHM	10
	5.1 ARCHITECTURE DIAGRAM	10
6.	SYSTEM REQUIREMENTS	11
	6.1 HARDWARE REQUIREMENTS	11
	6.2 SOFTWARE REQUIREMENTS	11
7.	SOFTWARE ENVIRONMENT	12
	7.1 Python	12
	7.2 History of Python	12
	7.3 Python Features	13
	7.4 Getting Python	14

7.5FirstPythonProgram	15
8. Python Install	23
9. MODULES	27
10. MODULE DESCRIPTION	27
10.1 Data Collection Module	27
10.2 Preparing the data Module	27
10.3 Training a model	27
10.4 Disease prediction Module	28
11. DATA FLOW DIAGRAM	29
12. ER DIAGRAM	31
13. UML DIAGRAM	32
14. CLASS DIAGRAM	33
15. ACTIVITY DIAGRAM	34
16. SEQUENCE DIAGRAM	35
17. COLLABARATION DIAGRAM	36
18. INPUT DESIGN AND OUTPUT DESIGN	37
18.1 INPUT DESIGN	37
18.2 OBJECTIVES	37
18.3 OUTPUT DESIGN	38
19. SYSTEM STUDY	40
19.1 FEASIBILITY STUDY	40
19.2 SYSTEM TESTING	42
19.2.1 TYPES OF TESTING	42
<i>SOURCE CODE</i>	48
<i>SCREEN SHOTS</i>	60

List of Figures

Fig No.	Title	Page No.
5.1	<i>ARCHITECTURE DIAGRAM</i>	10
11.1	<i>DATA FLOW DIAGRAM</i>	30
12.1	<i>ER DIAGRAM</i>	31
13.1	<i>UML DIAGRAM</i>	32
14.1	<i>CLASS DIAGRAM</i>	33
15.1	<i>ACTIVITY DIAGRAM</i>	34
16.1	<i>SEQUENCE DIAGRAM</i>	35
17.1	<i>COLLABARATION DIAGRAM</i>	36

1.INTRODUCTION

- The Earth is going through a purplish patch of technology where the demand of intelligence and accuracy is increasing behind it. Today's people are likely addicted to internet but they are not concerned about their physical health.
- People ignore the small problem and don't visit to visit hospital which turn into serious disease with time. Taking the advantage of this growing technology, our basis aim is to develop such a system that will predict the multiple diseases in accordance with symptoms put down by the patients without visiting the hospitals / physicians.
- Machine Learning is a subset of AI that is mainly deal with the study of algorithms which improve with the use of data and experience. Machine Learning has two phases i.e. Training and Testing. Machine Learning provides an efficient platform in medical field to solve various healthcare issues at a much faster rate.
- There are two kinds of Machine Learning – Supervised Learning and Unsupervised Learning. In supervised learning we frame a model with the help of data that is well labelled. On the other hand, unsupervised learning model learn from unlabeled data.
- The intent is to deduce a satisfactory Machine Learning algorithm which is efficient and accurate for the prediction of disease. In this paper, the supervised Machine Learning concept is used for predicting the diseases.
- The main feature will be Machine Learning in which we will be using machine learning algorithm which will help in early prediction of diseases accurately and better patient care.

1.1 PROBLEM DEFINITION

The smart health prediction system focused for optimally reducing the healthcare costs. There are several functionalities remain untouched into health prediction system. So by living in the edge of technology and still if we are not able to utilize it in efficient and proper manner then there is no use of it. To tackle this, research is carried out in health prediction system. There are several applications which use any one of the technology. This project shows the merging of both technologies to achieve efficient result.

1.2 OBJECTIVES

1.2.1 Aim of the Project:

- The analysis accuracy is reduced when the quality of medical data is incomplete.
- Moreover, different regions exhibit unique characteristics of certain regional diseases, which may weaken the prediction of disease outbreaks.
- However, those existing work mostly considered structured data.
- There is no proper methods to handle semi structured and unstructured.
- The proposed system will consider both structured and unstructured data.
- The analysis accuracy is increased by using Machine Learning algorithm.

1.2.2 Scope of the Project

- In this work, our goal is to provide a tool to assist professionals and consumers in finding and choosing disease.
- To achieve this goal, we develop an approach that allows a user to query for disease that satisfy a set of conditions based on disease properties, such as disease indications and also takes into account patient profiles.

2 LITERATURE SURVEY

Machine learning-based method for personalized and cost-effective detection of Alzheimer's disease

Diagnosis of Alzheimer's disease is often difficult, especially early in the disease process at the stage of mild cognitive impairment. Yet, it is at this stage that treatment is most likely to be effective, so there would be great advantages in improving the diagnosis process. We describe and test a machine learning approach for personalized and cost-effective diagnosis of AD. It uses locally weighted learning to tailor a classifier model to each patient and computes the sequence of biomarkers most informative or cost-effective to diagnose patients. Using ADNI data, we classified AD versus controls and MCI patients who progressed to AD within a year, against those who did not. The approach performed similarly to considering all data at once, while significantly reducing the number (and cost) of the biomarkers needed to achieve a confident diagnosis for each patient. Thus, it may contribute to a personalized and effective detection of AD, and may prove useful in clinical settings.

Effect of Meteorological Conditions on Occurrence of Hand, Foot and Mouth Disease in Wuwei City, Northwestern China

The main objective of this paper is to supply scientific basics for preventing and forecasting the prevalence of hand, foot and mouth disease to explore the effect of different meteorological conditions on occurrence of hand, foot and mouth disease in Wuwei City, northwestern China. Here the data about the diseases and weather was collected from 2008-2010, and the correlation analysis, multiple linear regression and exponential curve fitting methods were made. The results showed

that 2688 cases of hand, foot and mouth disease were collected from 2008 to 2010, and the annual average incidence was 47.62/100,000. The average prevalence of hand, foot and mouth disease at Liangzhou District, Minqin County, Gulang County and Tianzhu Tibetan Autonomous County were 42.69, 38.52, 65.92 and 49.18 per 100,000 respectively. This disease occurred year-round in Wuwei City, but had a clear seasonal climax. Generally, the incidence increased from April and rose to the first peak in May, Jun, July respectively. The second peak was in September or October every year. Different meteorological factors had different impact on the epidemic of disease in four areas, such as average temperature, relative humidity, atmospheric pressure, rainfall and evaporation capacity. The results of multiple linear regressions indicated that relative humidity and atmospheric pressure were the main influence factors in Liangzhou District, average temperature in Gulang County, atmospheric pressure in Tianzhu County. The incidence of the disease and average sunshine hours showed exponential function relationship in Minqin County. In conclusion, different weather conditions have different impact on the prevalence of hand, foot and mouth disease. A high correlation exists in four areas of Wuwei City between meteorological factors and hand, foot and mouth disease occurrence. And summer and autumn were the important seasons to prevent and control the disease.

Developing an Index for Detection and Identification of Disease Stages

Spectral data have been widely used to estimate the disease severity levels of different plants. However, such data have not been evaluated to estimate the disease stages of the plant. This study aimed at developing a spectral disease index that is able to identify the stages of wheat leaf rust disease at various DS levels. To meet the aim of the study, the reflectance spectra of infected leaves with different

symptom fractions and DS levels were measured with a spectroradiometer. Then, pure spectra of the different disease symptoms at the leaf scale were analyzed, and a new function was developed to find the wavelengths most sensitive to disease symptom fraction. The reflectance spectra with highest sensitivity were found at 675 and 775 nm. Finally, the normalized difference of DS and the ratio ρ_{675}/ρ_{775} was used as a new SDI to discriminate three different levels of the disease stage at the canopy level. The suggested SDI showed a promising performance to improve the detection disease stages in precision plant protection.

Quantized Analysis for Heart Valve Disease based on Cardiac Sound Characteristic Waveform Method

In order to analyze heart valve disease accurately and effectively, a new quantized diagnosis method was proposed to analyze four clinical heart valve sounds, namely cardiac sound characteristic waveform. BIOPAC acquiring system was used to collect signal. The recorded data is transmitted to a computer by ethernet for storage ! !analysis and display in real-time. Analytical model of single degree-of-freedom was established to extract characteristic waveform. Furthermore, diagnosis parameters were calculated to discriminate heart sound of normal and heart valve disease by easy-understanding graphical representation, so that, even for an inexperienced user is able to monitor his or her pathology progress easily. Finally, a case study on a heart valve disease patient before and after surgery is demonstrated to validate the usefulness and efficiency of the proposed method.

Non-Linear Analysis of Heart Rate Variability in Patients with Coronary Heart Disease

The article emphasizes clinical and prognostic significance of non-linear measures of the heart rate variability, applied on the group of patients with coronary heart disease and age-matched healthy control group. Three different methods were applied: Hurst exponent, Detrended Fluctuation Analysis and approximate entropy. Hurst exponent of the R-R series was determined by the range rescaled analysis technique. DFA was used to quantify fractal long-range-correlation properties of heart rate variability. Approximate entropy measures the unpredictability of fluctuations in a time series. It was found that the short-term fractal scaling exponent. The patients with CHD had lower Hurst exponent in each program of exercise test separately, as well as approximate entropy than healthy control group.

3. EXISTING SYSTEM

Machine can predict diseases but cannot predict the sub types of the diseases caused by occurrence of one disease. It fails to predict all possible conditions of the people. Existing system handles only structured data. The prediction system are broad and ambiguous. In current past, countless disease estimate classifications have been advanced and in procedure. The standing organizations arrange a blend of machine learning algorithms which are judiciously exact in envisaging diseases. However the restraint with the prevailing systems are speckled. First, the prevailing systems are dearer only rich people could pay for to such calculation systems. And also, when it comes to folks, it becomes even higher. Second, the guess systems are non-specific and indefinite so far. So that, a machine can envisage a positive disease but cannot expect the sub types of the diseases and diseases caused by the existence of one bug. For occurrence, if a group of people are foreseen with Diabetes, doubtless some of them might have complex risk for Heart viruses due to the actuality of Diabetes. The remaining schemes fail to foretell all possible surroundings of the tolerant.

3.1 DISADVANTGES OF EXISTING SYSTEM

- Does not analyze the disease
- Less security
- There is no feedback system

4. PROPOSED SYSTEM

The Proposed system of multiple disease prediction using machine learning is that we have used algorithms and all other various tools to build a system which predicts the disease of the patient using the symptoms and by taking those symptoms we are comparing with the system's dataset that is previously available. By taking those datasets and comparing with the patient's disease we will predict the accurate percentage disease of the patient. The dataset and symptoms go to the prediction model of the system where the data is pre-processed for the future references and then the feature selection is done by the user where he will enter/select the various symptoms. Then the classification of those data is done with the help of machine learning algorithms such as Logistic regression. Then the data goes in the recommendation model, there it shows the risk analysis that is involved in the system and it also provides the probability estimation of the system such that it shows the various probability like how the system behaves when there are n number of predictions are done and it also does the recommendations for the patients from their final result and also from their symptoms like it can show what to use and what not to use from the given datasets and the final results. It predicts probable diseases by mining data sets such as Covid-19, Chronic Kidney disease and heart Disease. To the best of our knowledge in the area of medical big data analytics none of the existing work focused on both data types.

4.1 ADVANTAGES OF PROPOSED SYSTEM

- Easily analyze the disease
- High Accuracy

5. PROPOSED ALGORITHM

- Logistic Regression

5.1 ARCHITECTURE DIAGRAM

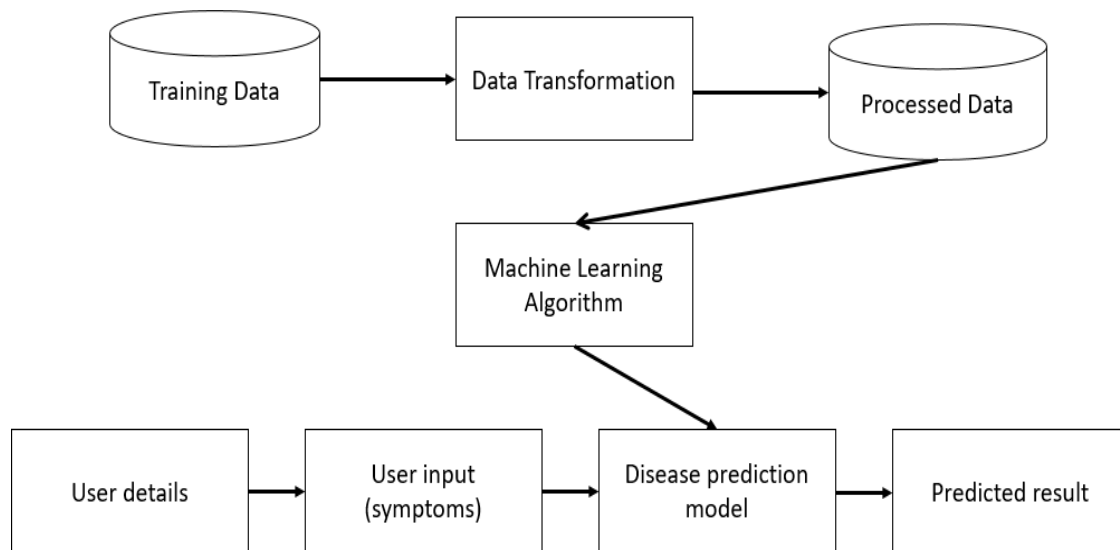


Fig 5.1- ARCHITECTRE DIAGRAM

6. SYSTEM REQUIREMENTS

6.1 HARDWARE REQUIREMENTS

- Processor : Core i3/i5/i7
- RAM : 2-4GB
- HDD : 500 GB

6.2 SOFTWARE REQUIREMENTS

- Platform : Windows Xp/7/8/10
- Coding Language : Python

7.

SOFTWARE ENVIRONMENT

7.1 Python:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

7.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

7.3 Python Features

Python's features include –

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.

- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

7.4 Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python <https://www.python.org>.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to <https://www.python.org/downloads/>.

- Follow the link for the Windows installer python-XYZ.msifile where XYZ is the version you need to install.
- To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

7.5 First Python Program

Let us execute programs in different modes of programming.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

```
$ python
Python2.4.3(#1,Nov112010,13:34:43)
[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter –

```
>>>print"Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!")**; However in Python version 2.4.3, this produces the following result –

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file –

```
print"Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows –

```
$ python test.py
```

This produces the following result –

```
Hello, Python!
```

Flask Framework:

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods –

Sr.No	Methods & Description
1	GET Sends data in unencrypted form to the server. Most common method.
2	HEAD Same as GET, but without response body
3	POST Used to send HTML form data to server. Data received by POST method is not cached by server.
4	PUT Replaces all current representations of the target resource with the uploaded content.

5

DELETE

Removes all current representations of the target resource given by a URL

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Save the following script as login.html

```
<html>

<body>

<form action="http://localhost:5000/login" method="post">

<p>Enter Name:</p>

<p><input type="text" name="nm"/></p>

<p><input type="submit" value="submit"/></p>

</form>

</body>

</html>
```

Now enter the following script in Python shell.

```
from flask import Flask, redirect, url_for, request
```

```
app=Flask(__name__)

@app.route('/success/<name>')

def success(name):

return'welcome %s'% name

@app.route('/login',methods=['POST','GET'])

def login():

if request.method=='POST':

user=request.form['nm']

return redirect(url_for('success',name= user))

else:

user=request.args.get('nm')

return redirect(url_for('success',name= user))

if __name__=='__main__':

app.run(debug =True)
```

After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**.

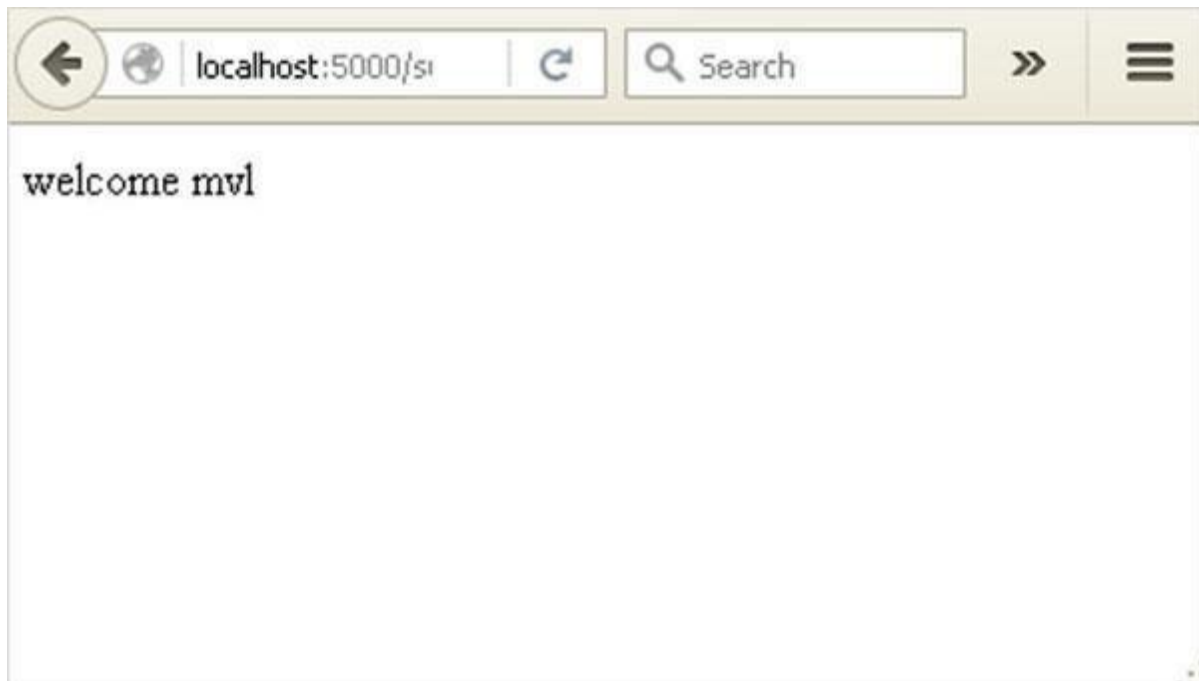
A screenshot of a web browser window. The address bar shows the file path 'file:///C:/login.ht'. The main content area displays a simple login form. It starts with the text 'Enter Name:' followed by a text input field containing the value 'mvl'. Below the input field is a button labeled 'submit'.

Form data is POSTed to the URL in action clause of form tag.

http://localhost/login is mapped to the **login()** function. Since the server has received data by **POST** method, value of „nm“ parameter obtained from the form data is obtained by –

```
user = request.form['nm']
```

It is passed to „**success**“ URL as variable part. The browser displays a **welcome** message in the window.



Change the method parameter to „**GET**“ in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of „nm“ parameter is now obtained by –

```
User = request.args.get(„nm“)
```

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to „nm“ parameter is passed on to „/success“ URL as before.

What is Python?

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

8. Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

```
helloworld.py
```

```
print("Hello, World!")
```

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

```
C:\Users\Your Name>python
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
```

Which will write "Hello, World!" in the command line:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
```

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

```
exit()
```

Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

```
>>>                                     print("Hello,                               World!")
Hello, World!
```

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

```
C:\Users\Your Name>python myfile.py
```

Python Indentations

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important.

Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
```

```
    print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

```
if 5 > 2:
```

```
print("Five is greater than two!")
```

Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

Example

Comments in Python:

```
#This is a comment.
```

```
print("Hello, World!")
```

Docstrings

Python also has extended documentation capability, called docstrings.

Docstrings can be one line, or multiline.

Python uses triple quotes at the beginning and end of the docstring:

Example

Docstrings are also comments:

```
"""This is a  
multiline docstring."""
```

```
print("Hello, World!")
```

9. MODULES

- Data Collection Module
- Preparing the data Module
- Training a model
- Disease prediction Module

10. MODULE DESCRIPTIONS

10.1 Data Collection Module

Be it the raw data from excel, access, text files etc., this step (gathering past data) forms the foundation of the future learning. The better the variety, density and volume of relevant data, better the learning prospects for the machine becomes.

10.2 Preparing the data Module

Any analytical process thrives on the quality of the data used. One needs to spend time determining the quality of data and then taking steps for fixing issues such as missing data and treatment of outliers. Exploratory analysis is perhaps one method to study the nuances of the data in details thereby burgeoning the nutritional content.

10.3 Training a model

This step involves choosing the appropriate algorithm and representation of data in the form of the model. The cleaned data is split into two parts – train and test

(proportion depending on the prerequisites); the first part (training data) is used for developing the model. The second part (test data), is used as a reference.

10.4 Disease prediction Module

Patient will specify the symptoms caused due to his illness. System will ask certain question regarding his illness and system predict the disease based on the symptoms specified by the patient.

11. DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

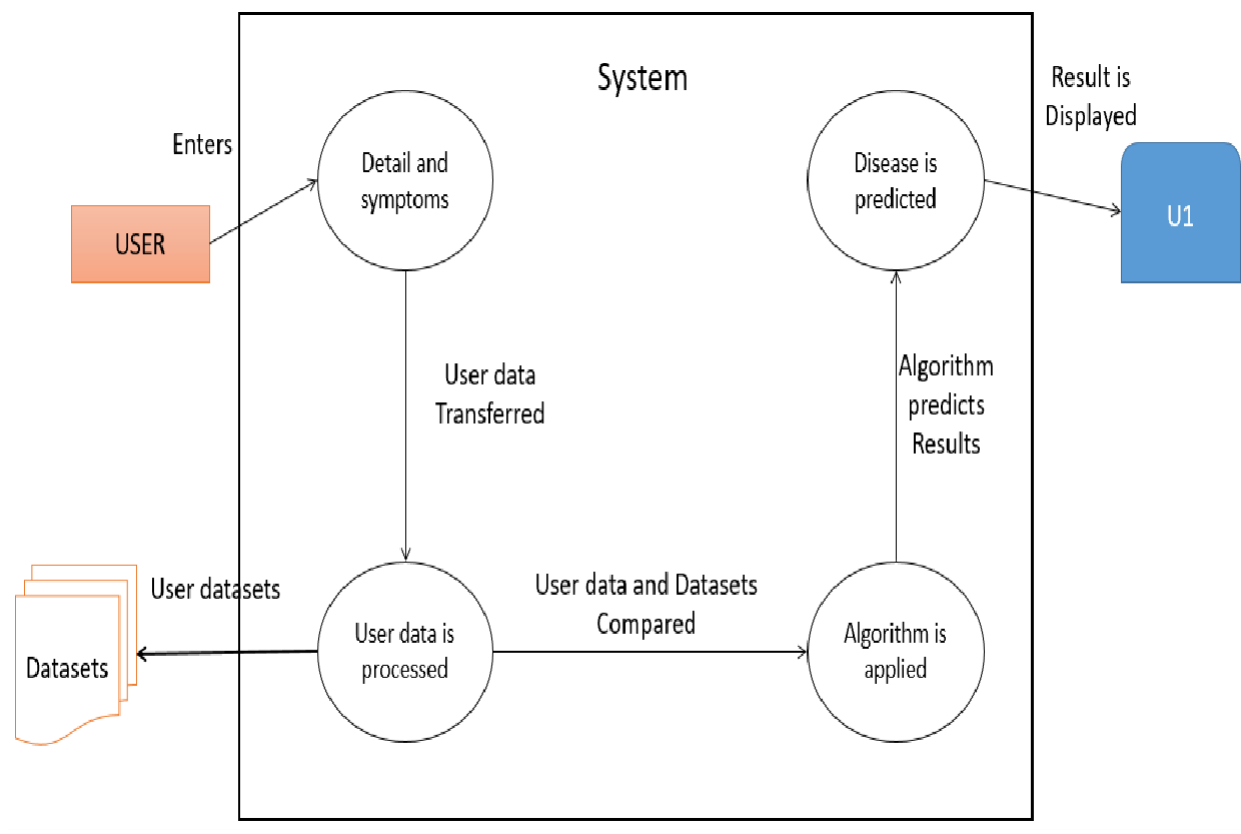


Fig 11.1- Data flow Diagram

12. ER- DIAGRAM

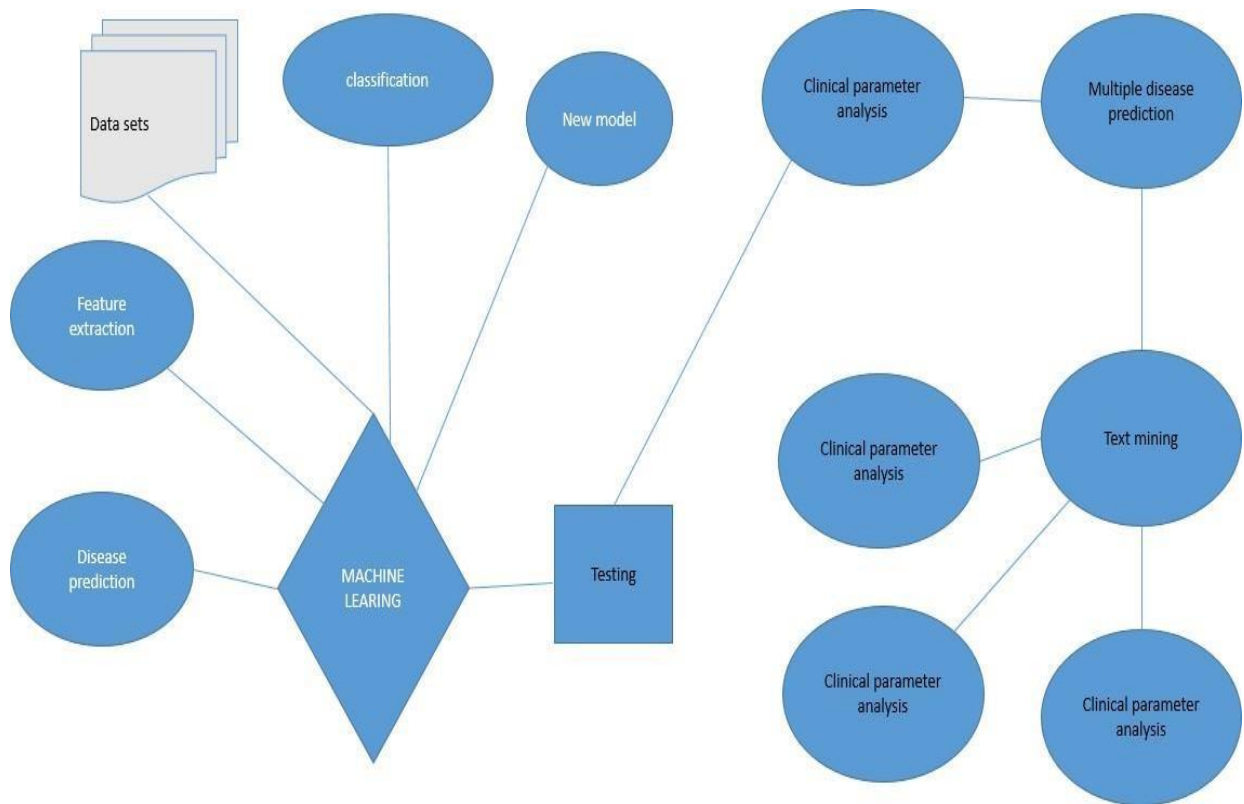


Fig 12.1- ER-Diagram

13. UML DIAGRAM

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. UML is a very important part of developing objects oriented software and the software development process. UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.



Fig 13.1- UML Diagram

14. CLASS DIAGRAM

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. These perspectives become evident as the diagram is created and help solidify the design. Class diagrams are arguably the most used UML diagram type. It is the main building block of any object oriented solution. It shows the classes in a system, attributes and operations of each class and the relationship between each class. In most modeling tools a class has three parts, name at the top, attributes in the middle and operations or methods at the bottom. In large systems with many classes related classes are grouped together to create class diagrams. Different relationships between diagrams are show by different types of Arrows. Below is a image of a class diagram. Follow the link for more class diagram examples.

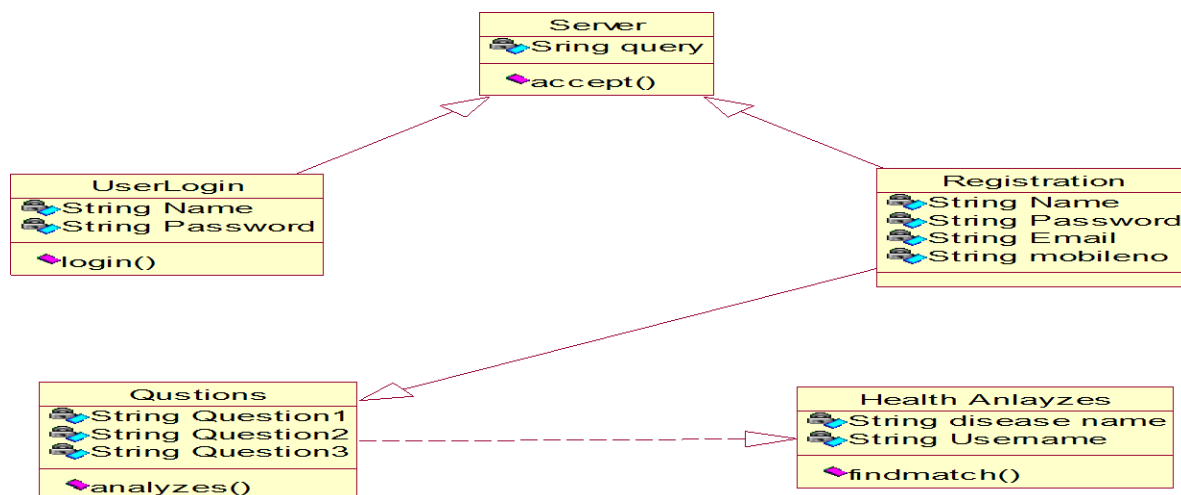


Fig 14.1- Class Diagram

15. ACTIVITY DIAGRAM

Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel.

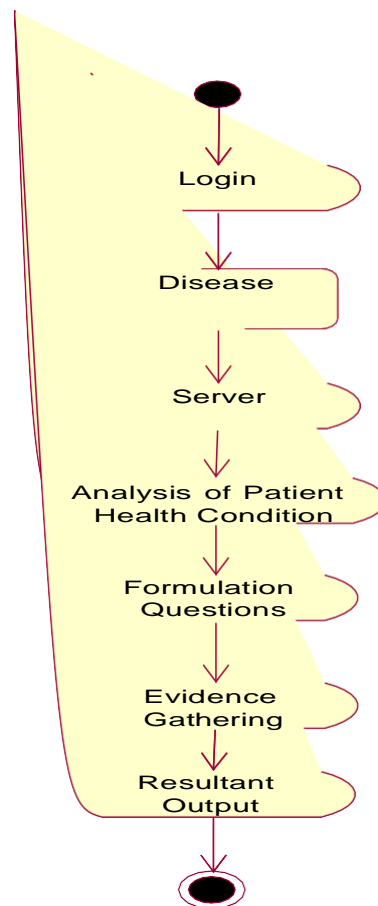


Fig 15.1- Activity Diagram

16. SEQUENCE DIAGRAM

Sequence diagrams in UML shows how object interact with each other and the order those interactions occur. It's important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are show as arrows. This article explains the purpose and the basics of Sequence diagrams.

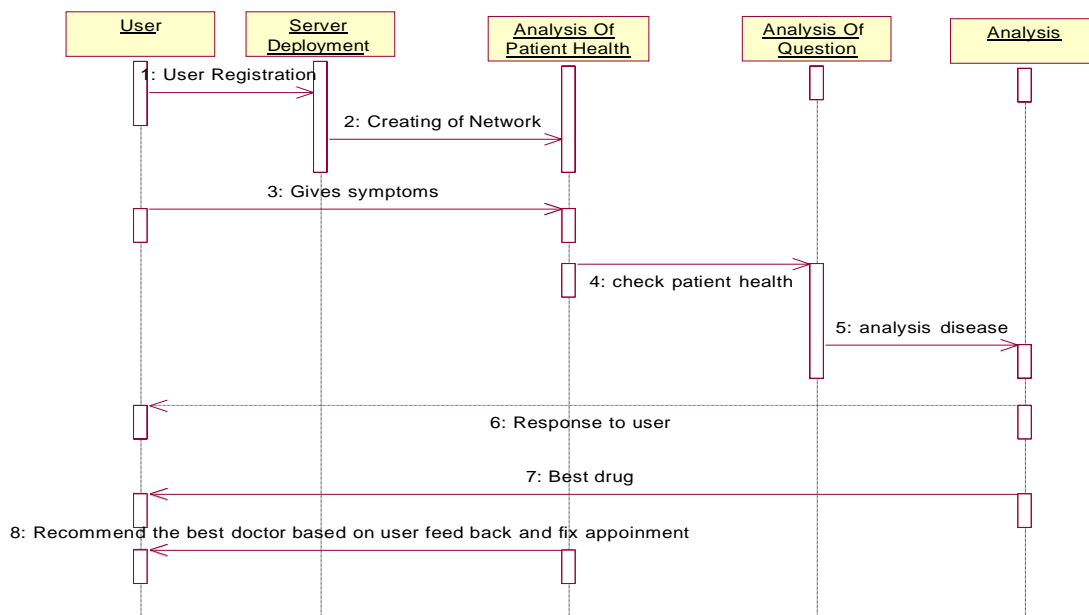


Fig 16.1- Sequence Diagram

17. COLLABORATION DIAGRAM

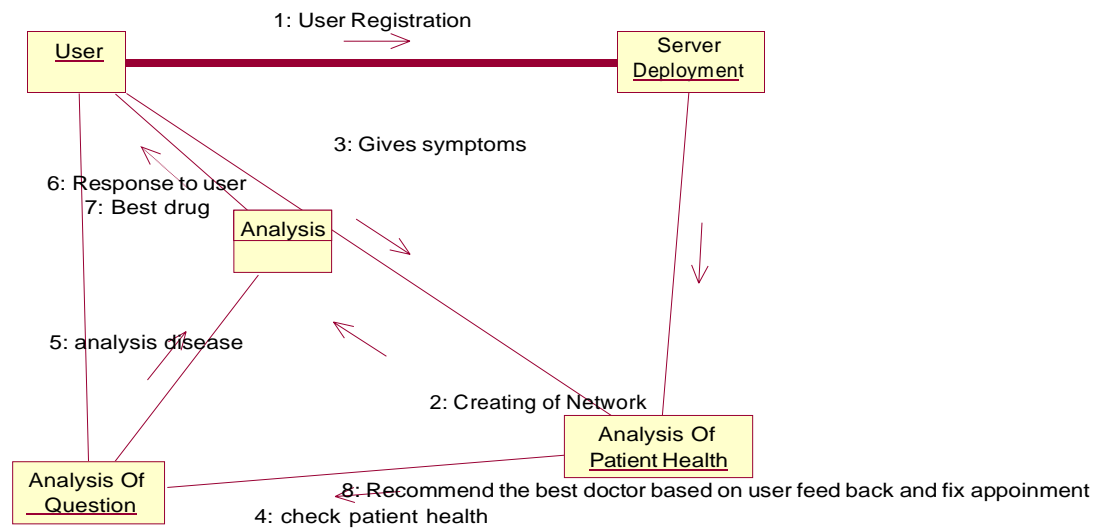


Fig 17.1- Collaboration Diagram

18.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

18.2 OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data

input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

18.3 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and

effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- ❖ Convey information about past activities, current status or projections of the
- ❖ Future.
- ❖ Signal important events, opportunities, problems, or warnings.
- ❖ Trigger an action.
- ❖ Confirm an action.

19. SYSTEM STUDY

19.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

19.2 SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

19.2.1 TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process

performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

SOURCE CODE

```
from flask import Flask, render_template, request, url_for, send_file, flash,
redirect, make_response
import pickle
import numpy as np
import os
import json
import termcolor
import smtplib

import CurrentStats
# import CancerModel
# import PdfConverter
# from PdfConverter import PDFPageCountError
import DiseasePred

# import warnings

app = Flask(__name__)
app.config['SECRET_KEY'] = '73a4b6ca8cb647a20b71423e31492452'

# For Coronavirus
with open("Coronavirus_logistic", "rb") as f:
    logisticRegression = pickle.load(f)

# For Chronic kidney disease
```

```

with open("CKD_Model", "rb") as f:
    decisionTree = pickle.load(f)

# For Heart Disease
with open("HeartDisease", "rb") as f:
    randomForest = pickle.load(f)

@app.route("/")
@app.route("/home")
def Homepage():
    # cases, cured, death = CurrentStats.currentStatus()
    return render_template("Homepage.html", feedback="False")

@app.errorhandler(404)
def page_not_found(e):
    return render_template("PageNotFound.html")

# @app.route("/currentstats", methods=["POST", "GET"])
# def CurrentStatus():
#     cases, cured, death = CurrentStats.currentStatus()
#     scases = scured = sdeath = 0
#     state = ""
#     try:
#         if request.method == "POST":

```

```

#         # print(request.form)
#         formDict = request.form
#         state = formDict['state']
#         # print(state)
#         scases, scured, sdeath = CurrentStats.StateStatus(state)
#     except UnboundLocalError:
#         flash("The State is not Affected Yet")
#     except ValueError:
#         flash("The State is not Affected Yet")
#     return render_template("CurrentStats.html", state=state, scases=scases,
scured=scured, sdeath=sdeath, cases=cases, cured=cured, death=death,
title="Current          Statistics",          navTitle="Current          Status",
headText="Coronavirus          Current          Stats          Statewise",
ImagePath="/static/Virus.png")

```

```

@app.route("/about")

```

```

def About():

```

```

    return render_template("About.html")

```

```

@app.route("/contact", methods=["POST", "GET"])

```

```

def Contact():

```

```

    if request.method == "POST":

```

```

        # print(request.form)

```

```

        contactDict = request.form

```

```

        firstname = contactDict['firstname']

```

```

lastname = contactDict['lastname']
email = contactDict['email']
phone = int(contactDict['phone'])
description = contactDict['description']

subject = "Medical Website feedback !!"
message = f"First Name : {firstname} \nLast Name : {lastname} \nEmail :
{email}\nPhone Number : {phone}\nDescription : {description}\n"
content = f"Subject : {subject} \n\n{message}"
sender = "intmain1221@gmail.com"
receiver = "intmain1221@gmail.com"
password = "intmain@11"

print(content)
try:
    with smtplib.SMTP('smtp.gmail.com', 587) as mail:
        mail.ehlo()
        mail.starttls()
        mail.login(sender, password)
        mail.ehlo()
        mail.sendmail(sender, receiver, content)

print("Mail Send Successfully !")
cases, cured, death = CurrentStats.currentStatus()
return render_template("Homepage.html", cases=cases, cured=cured,
death=death, feedback="True")

```

```

    except:
        pass
    return render_template("Contact.html")

@app.route("/infected")
def Infected():
    return render_template("Infected.htm", disease="Nothing")

@app.route("/noninfected")
def NonInfected():
    return render_template("NonInfected.htm")

@app.route("/download")
def Download():
    file = "static/Example.docx"
    return send_file(file, as_attachment=True)

@app.route("/HeartDisease", methods=["POST", "GET"])
def Heart_disease():
    if request.method == "POST":
        # print(request.form)
        heart_dict = request.form

```

```

age = int(heart_dict['age'])
gender = int(heart_dict['gender'])
height = int(heart_dict['height'])
weight = int(heart_dict['weight'])
sbp = int(heart_dict['sbp'])
dbp = int(heart_dict['dbp'])
cholesterol = int(heart_dict['cholesterol'])
glucose = int(heart_dict['glucose'])
smoke = int(heart_dict['smoke'])
alcohol = int(heart_dict['alcohol'])
active = int(heart_dict['active'])
age = age*365
model_input = [age, gender, height, weight, sbp,
               dbp, cholesterol, glucose, smoke, alcohol, active]
prediction = randomForest.predict([model_input])[0]

if prediction:
    return render_template("Infected.htm", disease="Heart Disease")
else:
    return render_template("NonInfected.htm")

return render_template("HeartDisease.html", title="Heart Disease
Detector", navTitle="Heart Disease Detector", headText="Heart Disease
Probabilty Detector", ImagePath="/static/HeartPulse.png")

@app.route("/diseaseprediction", methods=["POST", "GET"])

```

```

def Disease():
    symptoms = []
    if request.method == "POST":
        rf = request.form
        # print(rf)
        for key, value in rf.items():
            # print(key)
            symptoms.append(value)
        print(symptoms)

    if len(symptoms) < 5 or len(symptoms) > 8:
        flash("Please Select symptoms only between 5 and 8 Inclusive")
    else:
        prediction = DiseasePred.predicts(symptoms)
        if prediction:
            return render_template("Infected.htm", disease=prediction)
        else:
            return render_template("NonInfected.htm")
    return render_template("dp.html")

```

```

@app.route("/CKD", methods=["POST", "GET"])

```

```

def CKD():
    if request.method == "POST":
        submitted_values = request.form
        sg = str(float(submitted_values["sg"].strip()))
        albumin = str(float(submitted_values["albumin"].strip()))

```



```

    hemoglobin = str(float(submitted_values["hemoglobin"].strip()))
    pcv = str(float(submitted_values["pcv"].strip()))
    hypertension = str(float(submitted_values["hypertension"].strip()))
    sc = str(float(submitted_values["sc"].strip()))

    ckd_inputs1 = [sg, albumin, sc, hemoglobin, pcv, hypertension]
    prediction = decisionTree.predict([ckd_inputs1])
    # print("*****", prediction)
    if not prediction:
        return render_template("Infected.htm", disease="Chronic Kidney
Disease")
    else:
        return render_template("NonInfected.htm")

    return render_template("ChronicKidney.html", title="Chronic Kidney
Disease", navTitle="Chronic Kidney Disease", headText="Chronic Kidney
Disease Detector", ImagePath="/static/Chronic_Kidney.png")

@app.route("/CoronavirusPrediction", methods=["POST", "GET"])
def Coronavirus():
    if request.method == "POST":
        # print(request.form)
        submitted_values = request.form
        temperature = float(submitted_values["temperature"].strip())
        age = int(submitted_values["age"])
        cough = int(submitted_values["cough"])

```

```

cold = int(submitted_values["cold"])
sore_throat = int(submitted_values["sore_throat"])
body_pain = int(submitted_values["body_pain"])
fatigue = int(submitted_values["fatigue"])
headache = int(submitted_values["headache"])
diarrhea = int(submitted_values["diarrhea"])
difficult_breathing = int(submitted_values["difficult_breathing"])
travelled14 = int(submitted_values["travelled14"])
travel_covid = int(submitted_values["travel_covid"])
covid_contact = int(submitted_values["covid_contact"])

age = 2 if (age > 50 or age < 10) else 0
temperature = 1 if temperature > 98 else 0
difficult_breathing = 2 if difficult_breathing else 0
travelled14 = 3 if travelled14 else 0
travel_covid = 3 if travel_covid else 0
covid_contact = 3 if covid_contact else 0

model_inputs = [cough, cold, diarrhea,
                 sore_throat,    body_pain,    headache,    temperature,
difficult_breathing, fatigue, travelled14, travel_covid, covid_contact, age]
prediction = logisticRegression.predict([model_inputs])[0]
# print("*****", prediction)
if prediction:
    return render_template("Infected.htm", disease="Coronavirus")
else:
    return render_template("NonInfected.htm")

```

```
return render_template("Coronavirus.htm", title="Coronavirus  
Prediction", navTitle="COVID-19 Detector", headText="Coronavirus  
Probability Detector", ImagePath="/static/VirusImage.png")
```

```
if __name__ == '__main__':  
    app.run(threaded=True, debug=True)
```

MULTIPLE DISEASE PREDICTION

```
import pickle
```

```
import pandas as pd
```

```
def predicts(inputs):
```

```
    header = ['bloody_stools', 'fecal_leakage', 'swelling', 'dizziness', 'confusion',  
'fatigue', 'itching', 'vomiting', 'arm_pain',  
             'cough', 'muscle_pain', 'depression', 'fever',  
'painful_bowel_moments', 'urine_blood', 'sweating', 'nausea',  
             'stiff_neck', 'decreased_appetite', 'weak', 'wheezing', 'bleeding',  
'hives', 'bleed', 'headache', 'dry_mouth', 'sweat',  
             'stomach_pain', 'stool_pressure', 'anxiety', 'shoulder_pain',  
'anus_itching', 'vision_problem', 'abdominal_pain',  
             'chest_pain', 'weight_loss', 'diarrhea', 'breath_problems', 'thirsty',  
'anus_swelling', 'blood_o_tissue', 'constipation',
```

```
        'neck_pain',      'low_heartbeat',      'more_urine',      'low_breath',  
'muscle_cramps', 'muscle_spasm', 'yawning', 'rash', 'back_pain',  
        'anal_bleeding', 'lump_anus', 'cold', 'skin_rash', 'neck_stiff']
```

```
df = pd.read_csv("Disease_Symptoms.csv")
```

```
disease = set(df.iloc[:, 0])
```

```
disease = list(disease)
```

```
disease.sort()
```

```
model_inputs5 = []
```

```
for x in range(0, len(header)):
```

```
    model_inputs5.append(0)
```

```
# inputs = [i.strip() for i in input("Enter Symptoms : ").split()]
```

```
# print(inputs)
```

```
for element in range(0, len(header)):
```

```
    for symptom in inputs:
```

```
        if symptom == header[element]:
```

```
            model_inputs5[element] = 1
```

```
with open("DiseasePrediction(Dec)", "rb") as f:
```

```
    Model_Decision_Tree = pickle.load(f)
```

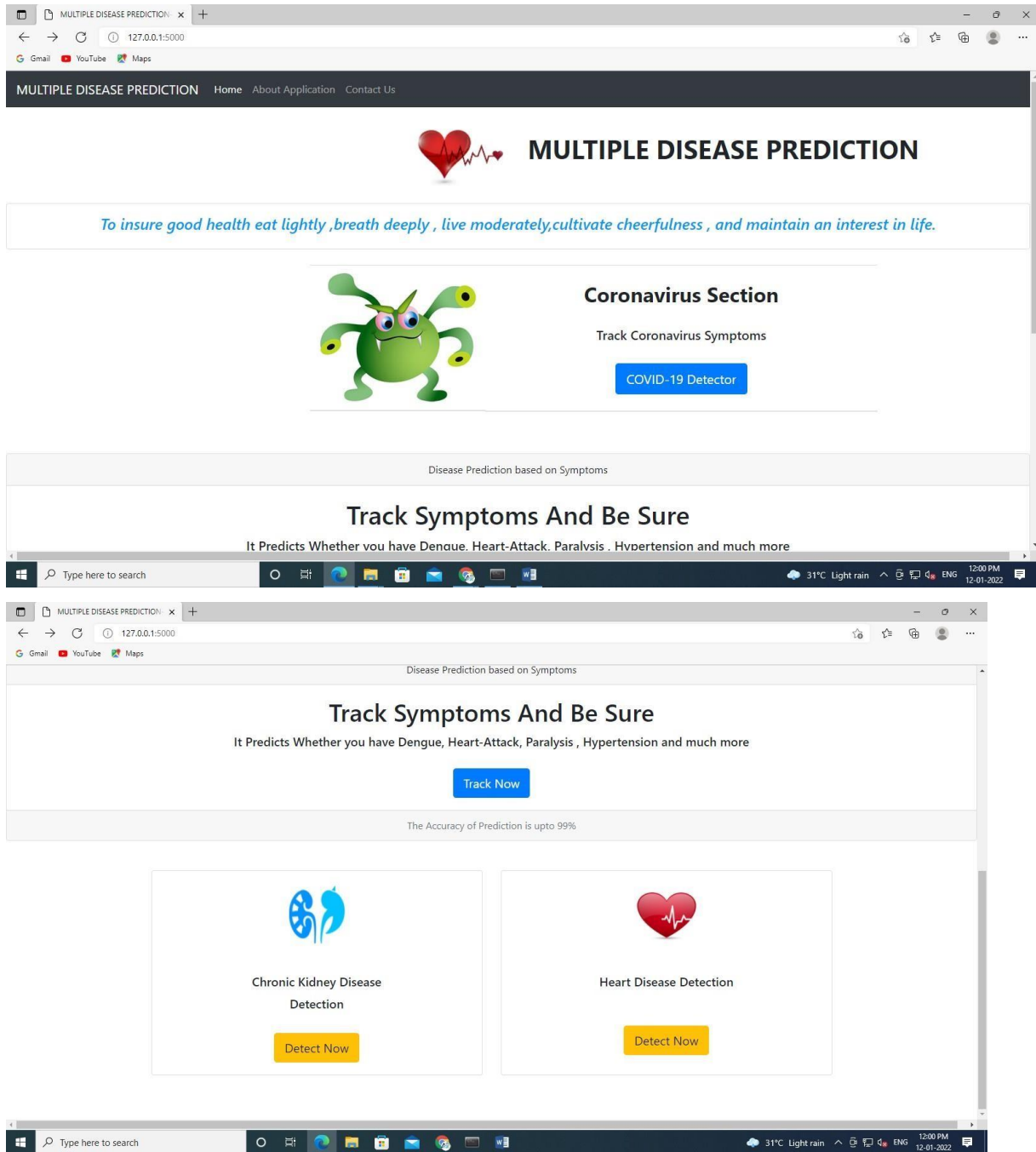
```
pred = Model_Decision_Tree.predict([model_inputs5])
```

```
print(disease[pred[0]])  
return disease[pred[0]]
```

```
if __name__ == '__main__':  
    # predicts(['fever', 'vomiting', 'headache', 'sweating',  
    #         'bloody_stools', 'abdominal_pain', 'diarrhea'])  
    Pass
```

SCREENSHOTS

Home Page



Covid-19 Detector


Safe Zone

Coronavirus Prediction

127.0.0.1:5000/CoronavirusPrediction

Gmail YouTube Maps

Coronavirus Probability Detector



Enter Body Temperature :
36.14

Age :
25

Cough : ☒ Yes ☐ No Cold : ☒ Yes ☐ No Sore Throat : ☒ Yes ☐ No

Body Pain : ☒ Yes ☐ No Fatigue : ☐ Yes ☒ No Headache : ☒ Yes ☐ No

Diarrhea : ☐ Yes ☒ No Difficulty in breathing : ☒ Yes ☐ No

Have you travelled recently during past 14 days :
No

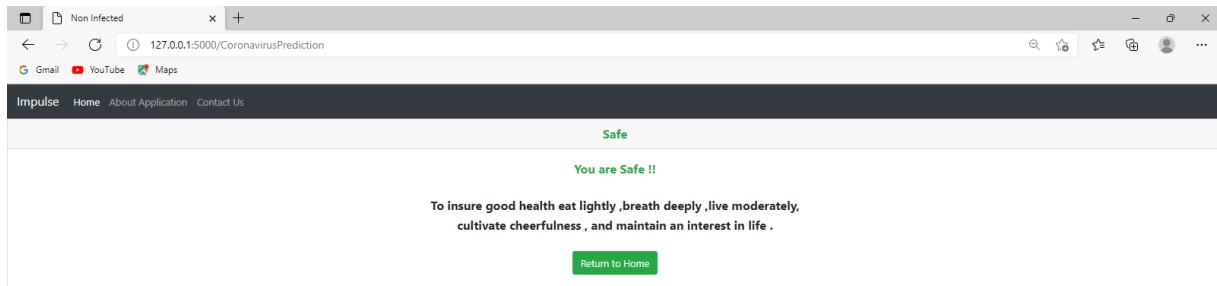
Do you have a travel history to a COVID-19 INFECTED AREA :
No

Do you have direct contact or is taking care of a positive COVID-19 PATIENT :
No

Submit

Type here to search

30°C Light rain 12:01 PM 12-01-2022



Danger Zone

A screenshot of a web browser window showing a form titled 'Coronavirus Probability Detector' with a cartoon virus character. The form contains several input fields and radio buttons for user information and symptoms. The inputs are: Body Temperature (47.34), Age (25), Cough (Yes), Cold (Yes), Sore Throat (Yes), Body Pain (Yes), Fatigue (No), Headache (Yes), Diarrhea (Yes), and Difficulty in breathing (No). There are three dropdown menus for travel history, all set to 'Yes'. A blue 'Submit' button is at the bottom.

Coronavirus Probability Detector

Enter Body Temperature :
47.34

Age :
25

Cough : ☒ Yes ☐ No Cold : ☒ Yes ☐ No Sore Throat : ☒ Yes ☐ No

Body Pain : ☒ Yes ☐ No Fatigue : ☐ Yes ☒ No Headache : ☒ Yes ☐ No

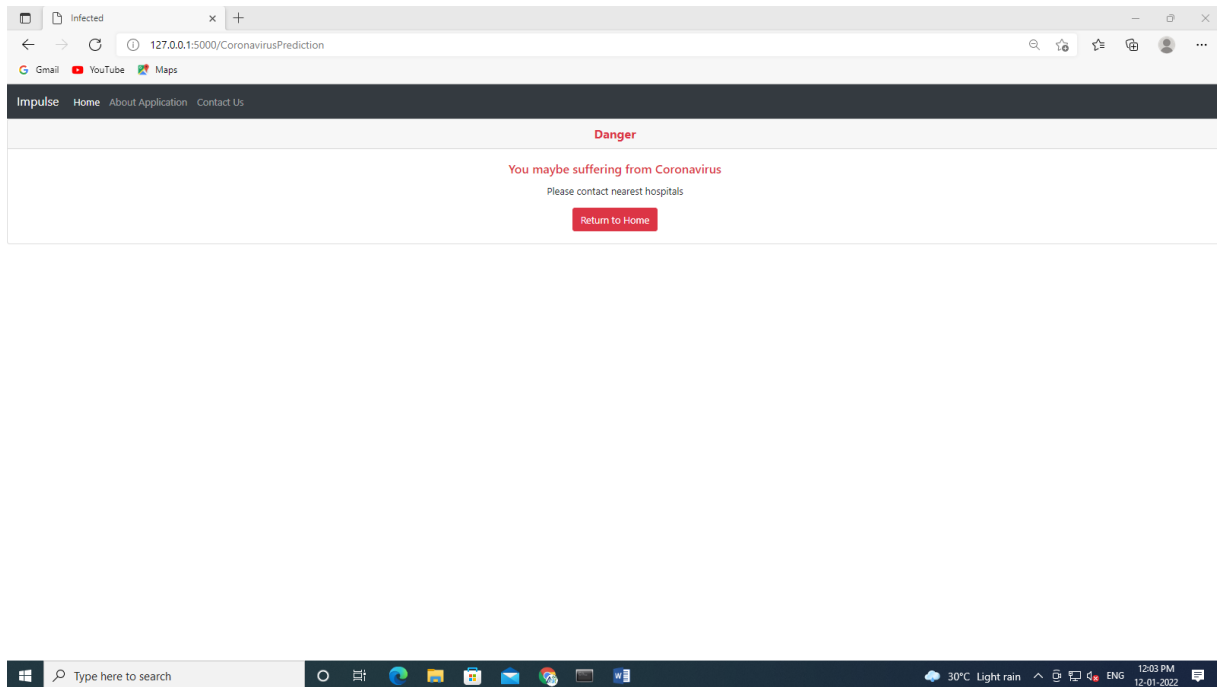
Diarrhea : ☒ Yes ☐ No Difficulty in breathing : ☒ Yes ☐ No

Have you travelled recently during past 14 days :
Yes

Do you have a travel history to a COVID-19 INFECTED AREA :
Yes

Do you have direct contact or is taking care of a positive COVID-19 PATIENT :
Yes

Submit



Multiple Disease Prediction Page

A screenshot of a web browser window showing a 'Medical Condition Prediction' form. The page has a light gray header with a logo and the title. The main content area has a heading 'Are you experiencing any of these symptoms below (mark all those applicable)'. There are two sections of symptoms, each with a 'Next' button. The first section includes: Fever (checked), Stomach Pain, Itching, Sweating, Cough (checked), Vision problem, Muscle Pain, Vomiting, Confusion, Depression, Shoulder pain, Cold, Arm pain, and Headache. The second section includes: Stiff Neck, Chest Pain, Skin rash, Back Pain (checked), Low heartbeat, Yawning, Swelling, Nausea, Low breath, Fatigue, Painful Bowel moments, Dry mouth, Stool pressure, and Urine Blood. The Windows taskbar at the bottom shows the search bar, taskbar icons, and system tray with weather and time information.

Disease Prediction

127.0.0.1:5000/diseaseprediction

☒ Back Pain ☐ Painful Bowel moments

Next

☐ Anal bleeding ☐ Constipation ☐ Bloody stools ☐ Fecal leakage
☐ Muscle cramps ☐ Abdominal Pain ☐ Weight loss ☐ Dizziness
☐ Sweat ☐ More Urine ☐ Bleed ☒ Neck pain
☐ Rash ☐ Muscle spasm ☒ Breath Problems

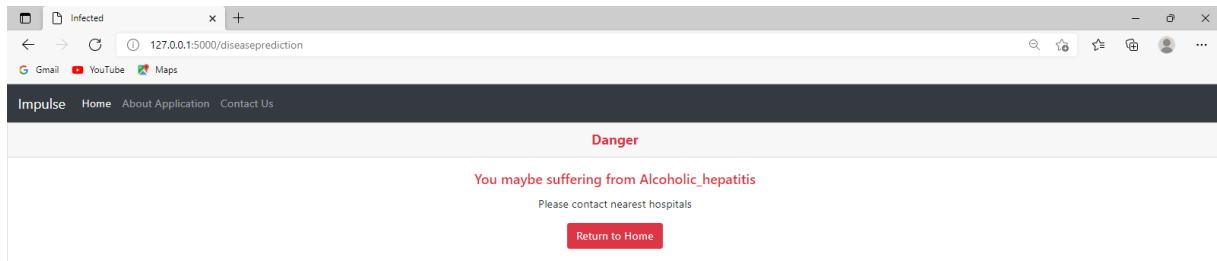
Next

☐ Anal swelling ☐ Thirsty ☐ Anus Itching ☐ Decreased appetite
☒ Wheezing ☐ Diarrhea ☐ Neck stiff ☐ Lump anus
☐ Anxiety ☐ More Urine ☐ Bleeding ☐ Hives

Submit

Type here to search

30°C Light rain 12:05 PM 12-01-2022



Pneumonia

A screenshot of a web browser window showing a 'Medical Condition Prediction' form. The browser address bar is '127.0.0.1:5000/diseaseprediction#fourth'. The form has a logo and the title 'Medical Condition Prediction'. Below it, the text asks 'Are you experiencing any of these symptoms below (mark all those applicable)'. The form contains two sections of checkboxes. The first section has checkboxes for Fever (checked), Stomach Pain, Itching, Sweating, Cough, Vision problem, Muscle Pain, Vomiting (checked), Confusion, Depression, Shoulder pain, Cold, Arm pain, and Headache (checked). A 'Next' button is below this section. The second section has checkboxes for Stiff Neck, Chest Pain, Skin rash, Back Pain, Low heartbeat, Yawning, Swelling, Nausea (checked), Low breath, Fatigue, Painful Bowel moments, Dry mouth, Stool pressure, and Urine Blood. The Windows taskbar is visible at the bottom.

Disease Prediction

127.0.0.1:5000/diseaseprediction#fourth

☐ Stiff Neck ☐ Low heartbeat ☒ Nausea ☐ Dry mouth

☐ Chest Pain ☐ Yawning ☐ Low breath ☐ Stool pressure

☐ Skin rash ☐ Swelling ☐ Fatigue ☐ Urine Blood

☐ Back Pain ☐ Painful Bowel moments

Next

☐ Anal bleeding ☐ Constipation ☒ Bloody stools ☐ Fecal leakage

☐ Muscle cramps ☒ Abdominal Pain ☐ Weight loss ☐ Dizziness

☐ Sweat ☐ More Urine ☐ Bleed ☐ Neck pain

☐ Rash ☒ Muscle spasm ☐ Breath Problems

Next

☐ Anal swelling ☐ Thirsty ☐ Anus Itching ☐ Decreased appetite

☐ Wheezing ☒ Diarrhea ☐ Neck stiff ☐ Lump anus

☐ Anxiety ☐ More Urine ☐ Bleeding ☐ Hives

Submit

Infected

127.0.0.1:5000/diseaseprediction

Impulse Home About Application Contact Us

Danger

You maybe suffering from Pneumonia

Please contact nearest hospitals

[Return to Home](#)

Chronic Kidney Disease Detection Page

Danger Zone

Chronic Kidney Disease Detector

Specific gravity : 0.75

Albumin : 0.2

Serum chistening : 0.033898

Hemoglobin : 0.836735

Packed cell volume : 0.717949

Hypertension : ☐ Yes ☒ No

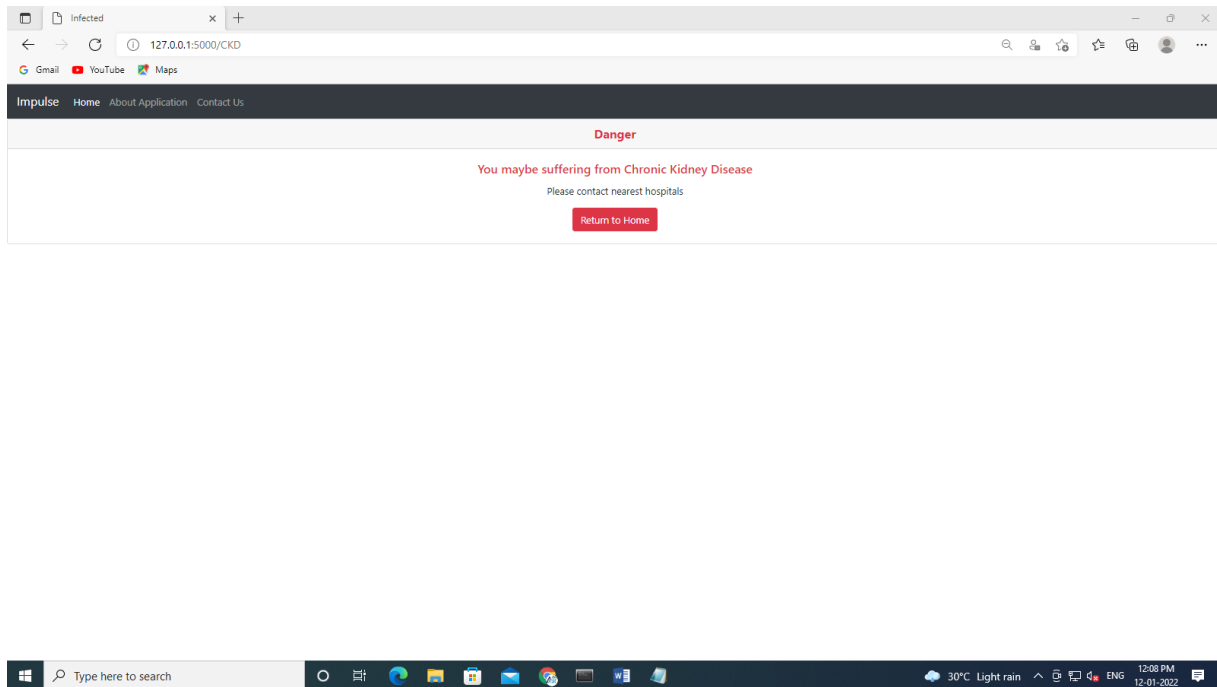
Submit

Some Facts about Chronic Kidney Disease

Deciding Factors

The survey data shows that the Hemoglobin , Packed cell volume , and Specific Gravity have 77% impact on deciding whether an individual is suffering from Chronic Kidney Disease or not

The Accuracy of the model is 98%



Safe Zone

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/CKD'. The page title is 'Chronic Kidney Disease Detector' with a kidney icon. The form contains the following fields and values:

Field	Value
Specific gravity :	1.020
Albumin :	0.0
Serum chritening :	1.2
Hemoglobin :	14.8
Packed cell volume :	30
Hypertension :	<input checked="" type="radio"/> Yes <input type="radio"/> No

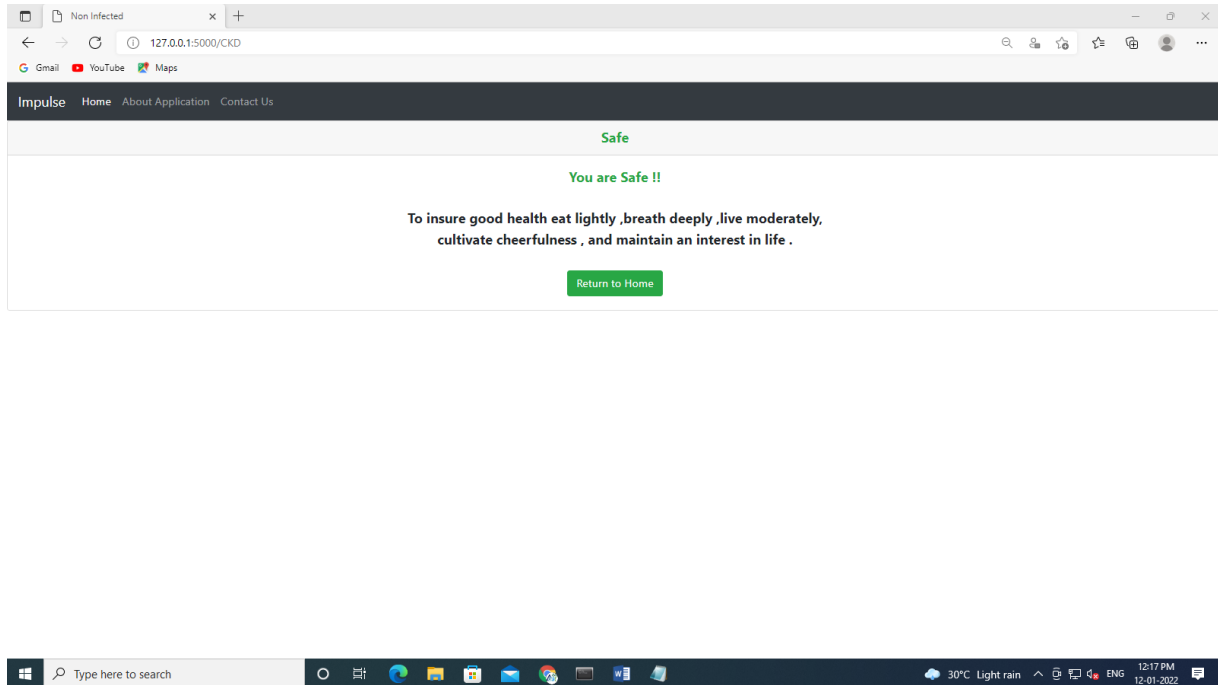
A blue 'Submit' button is located below the form. Below the button, a box titled 'Some Facts about Chronic Kidney Disease' contains the following text:

Deciding Factors

The survey data shows that the **Hemoglobin , Packed cell volume , and Specific Gravity** have 77% impact on deciding whether an individual is suffering from Chronic Kidney Disease or not


The Accuracy of the model is 98%

The Windows taskbar at the bottom shows the search bar, taskbar icons, and system tray with weather '30°C Light rain' and time '12:17 PM 12-01-2022'.



Heart Disease Detection Page

Safe Zone

Heart Disease Detector 

Age :

Gender : ☒ Male ☐ Female

Height : Weight :

Systolic blood pressure : Diastolic blood pressure :

Cholesterol : ☐ High ☒ Medium ☐ Low Glucose : ☐ High ☒ Medium ☐ Low

Do you smoke ?

Do you consume alcohol ?

Are you physically challenged ?

[Submit](#)

Non Infected

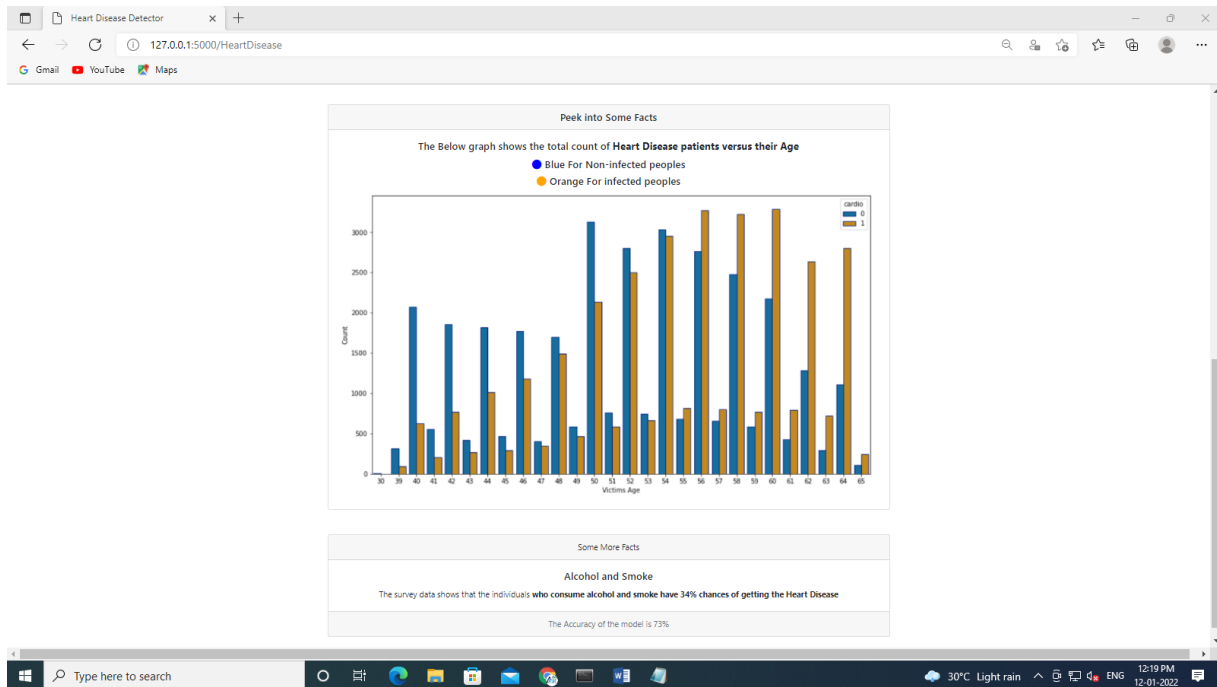
Impulse [Home](#) [About Application](#) [Contact Us](#)

Safe

You are Safe !!

To insure good health eat lightly ,breath deeply ,live moderately,
cultivate cheerfulness , and maintain an interest in life .

[Return to Home](#)



About Page

MULTIPLE DISEASE PREDICTION

Coronavirus Section

This Section Predicts whether an individual is been infected with Coronavirus or not

[Detect COVID-19](#)

Disease Prediction

We predict the following Diseases !

Malaria	Hypertension	Paralysis	Pneumonia
Dengue	Migraine	Drug Reaction	Dimorphic hemmorhoids(piles)
Heart Attack	Cervical spondylosis	Alcoholic hepatitis	

[Track Your Symptoms Now](#)


Special Predictions Center

Make Yourself Sure and Safe by Early Detection

The Following Models need Medical Report inorder to predict:

[Chronic Kidney Disease](#) [Heart Disease](#)

Danger Zone

Heart Disease Detector 

Age :

Gender : ☐ Male ☒ Female

Height : Weight :

Systolic blood pressure : Diastolic blood pressure :

Cholestrol : ☐ High ☒ Medium ☐ Low Glucose : ☐ High ☒ Medium ☐ Low

Do you smoke ?

Do you consume alcohol ?

Are you physically challenged ?

[Submit](#)

Impulse [Home](#) [About Application](#) [Contact Us](#)

Danger

You maybe suffering from Heart Disease

Please contact nearest hospitals

[Return to Home](#)

CONCLUSION

- This paper gives research of multiple researches done in this field. Our Proposed System aims at bridging gap between Doctors and Patients which will help both classes of users in achieving their goals.
- This system provides support for multiple disease prediction using different Machine Learning algorithms.
- The present approach of many systems focuses only on automating this process which lacks in building the user's trust in the system.
- By providing Doctor's recommendation in our system, we ensure user's trust side by side ensuring that the Doctor's will not feel that their Business is getting affected due to this System.

REFERENCES

- A. Singh et al., "Heart Disease Prediction Using Machine Learning Algorithms", *2020 International Conference on Electrical and Electronics Engineering (ICE3)*, pp. 452-457, February 2020.
- A Narin, C Kaya and Z. Pamuk, "Automatic detection of coronavirus disease (COVID-19) using x-ray images and deep convolutional neural networks", Mar 2020.
- Rajesh. Ranjan, "Predictions for COVID-19 outbreak in India using Epidemiological models", 2020.
- Mohan Senthilkumar, Chandrasegar Thirumalai and Gautam Srivastava, "Effective heart disease prediction using hybrid machine learning techniques", *IEEE Access*, vol. 7, pp. 81542-81554, 2019.
- Mohan Senthilkumar, Chandrasegar Thirumalai and Gautam Srivastava, "Effective heart disease prediction using hybrid machine learning techniques", *IEEE Access*, vol. 7, pp. 81542-81554, 2019.
- M. Bayati, S. Bhaskar and A. Montanari, "Statistical analysis of a low cost method for multiple disease prediction", *Statistical Methods Med. Res.*, vol. 27, no. 8, pp. 2312-2328, 2018.
- A. K. Shrivastava and S. Kumar Sahu, "Classification of Chronic Kidney Disease using Feature Selection Techniques", *IJCSE*, vol. 6, no. 5, pp. 649-653, 2018.

- N Chaithra and B Madhu, "Classification Models on Cardiovascular Disease Prediction using Data Mining Techniques", *Journal of Cardiovascular Diseases & Diagnosis*, vol. 6, pp. 1-4, 2018.