CHAPTER 1
INTRODUCTION

1.1GENERAL

The novel coronavirus disease 2019 (COVID-19) pandemic caused by the newly emerged SARS-CoV-2 is a critical and urgent threat to global health. The outbreak in early December 2019 in the Hubei province of the People’s Republic of China has spread worldwide. As of May 2020, the overall number of patients confirmed to have the disease has exceeded 3,580,000 in more than 180 countries, the number of people infected is probably much higher, and more than 250,000 people have died from COVID-19.

1.2 OBJECTIVE

The objective the project is prediction of COVID-19 by using the symptoms by various algorithms and finding the accuracy of those algorithms. And determining which one will provide the better accurate result among those algorithms.

1.3 Existing System:

K-Nearest Neighbors: In this method, K- Nearest Neighbors showed poor performance because KNN classifies test data directly from the dataset, no training was performed before testing.

Gaussian Naïve Bayes: At the training stage, it calculated the mean and standard deviation of each attribute. This mean and standard deviation were used to calculate the probabilities for the test data. For this reason, some attributes values are too big or too small from the mean. When testing data pattern contains those attributes values, it affects the classifier performance and sometimes gives wrong output label.

Logistic Regression: At the training stage, Logistic Regression algorithm estimated coefficient values by using stochastic gradient descent. The model can be trained for a fixed or as much as no of epochs by using stochastic gradient descent. Coefficients values are updated until the model predicts the correct class label for each training data.

LITERATURE SURVEY:

Title: An interactive web-based dashboard to track COVID-19 in real time

Author: Dong E

Year: 2020

Description:

In December, 2019, a local outbreak of pneumonia of initially unknown cause was detected in Wuhan (Hubei, China), and was quickly determined to be caused by a novel coronavirus,1 namely severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The outbreak has since spread to every province of mainland China as well as 27 other countries and regions, with more than 70 000 confirmed cases as of Feb 17, 2020.2 In response to this ongoing public health emergency, we developed an online interactive dashboard, hosted by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University, Baltimore, MD, USA, to visualise and track reported cases of coronavirus disease 2019 (COVID-19) in real time. The dashboard, first shared publicly on Jan 22, illustrates the location and number of confirmed COVID-19 cases, deaths, and recoveries for all affected countries. It was developed to provide researchers, public health authorities, and the general public with a user-friendly tool to track the outbreak as it unfolds. All data collected and displayed are made freely available, initially through Google Sheets and now through a GitHub repository, along with the feature layers of the dashboard, which are now included in the Esri Living Atlas.

1.4 Proposed System

Decision Tree (ID3): At training stage, it converted the continuous value's data into categorical values and given a range. When test data pattern contained values out of this given range, the classifier performance was affected and thus predicts wrong class label.

Random Forest: Random Forest is an ensemble classification method which is based on Decision Tree algorithm. This algorithm takes a portion of the dataset and then builds a tree, repeat this step for creating a forest by combining the generated trees. At the test stage, each tree predicts a class label for each test data and majority values of the class label is assigned to the test data. Therefore, it showed reasonable performance than conventional decision tree algorithm for this data.

SVM

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression.

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH).

- **Support Vectors −** Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.
- **Hyperplane −** As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

CHAPTER 2

PROJECT DESCRIPTION

2.1 GENERAL:

Predictions were generated using a gradient-boosting machine model built with decision-tree base-learners 10. Gradient boosting is widely considered state of the art in predicting tabular data 11 and is used by many successful algorithms in the field of machine learning 12. As suggested by previous studies 13, missing values were inherently handled by the gradient boosting predictor.

2.2 METHODOLOGIES

MODULE:

1. Data Source
2. Preprocessing
3. Splitting
4. Algorithms
5. Prediction

1. Data Source

The dataset contains initial records, on a daily basis, for all citizens tested for COVID19 nationwide. In addition to the test date and result, various information is available, including clinical symptoms, gender and a binary indication as to whether the tested individual is above age 60 years. Based on this data, we developed a model that predicts COVID-19 test results using eight features: gender, whether age is above 60, known contact with an infected individual, and five initial clinical symptoms.

2. Preprocessing

The real-life information contains large numbers with missing and noisy data. These data are pre-processed to overcome such issues and make predictions vigorously. Cleaning the collected data usually has noise and missing values. To get an accurate and effective result, the data need to be cleaned in terms of noise and missing values are to be filled up. Transformation it changes the format of the data from one form to another to make it more comprehensible. It involves smoothing, normalization, and aggregation tasks. Integration the data may not be acquired from a single source but varied sources, and it has to be integrated before processing. Reduction the data gained are complex and require to be formatted to achieve effective results.

3. Splitting

Splitting is an approach to protecting sensitive data from unauthorized access by encrypting the data and storing different portions of a file on different servers. The COVID-19 Symptoms data is splitting into both training and testing data for better prediction. The previous module introduced the idea of dividing your data set into two subsets:

- Training set—a subset to train a model.
- Test set—a subset to test the trained model.

4. Algorithms

Random forest is the ensemble learning approaches which constructs many decision trees at the time of training and predicts the output by analyzing individual trees. It is more efficient than decision tree. C4.5 uses the extension of the ID3 algorithm for generating decision trees for both nominal and numerical values. It is a statistical classifier which classifies as classes used in the diagnosis of certain patterns. A Random tree is a graphical representation of decisions with their respective consequences. A decision tree contains control statements, which are used to select the appropriate decision; the decision trees are mostly used in case of alternatives and possibilities.

5. Prediction

Prediction refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when you’re trying to forecast the likelihood of a particular outcome. The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be. Here we are predicting the COVID-19 Symptoms data by using various algorithms.

CHAPTER 3

REQUIREMENTS ENGINEERING

3.1 GENERAL

Effective screening enables quick and efficient diagnosis of COVID-19 and can mitigate the burden on healthcare systems. Prediction models that combine several features to estimate the risk of infection have been developed in hopes of assisting medical staff worldwide in triaging patients when allocating limited healthcare resources. These models use features such as computer tomography (CT) scans 2–5, information available at hospital admission including clinical symptoms 6, and laboratory tests.

3.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system does and not how it should be implemented.

• PROCESSOR	:	DUAL CORE 2 DUOS.
• RAM	:	4GB DD RAM
• HARD DISK	:	250 GB

3.3 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

SOFTWARE REQUIREMENTS

• Operating System	:	Windows 7/8/10
• Platform	:	Jupyter Notebook
• Programming Language	:	Python, HTML
• Front End	:	Jupyter Notebook

3.4 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute-based deduplication systems by resorting to the hybrid cloud architecture.

3.5 NON-FUNCTIONAL REQUIREMENTS

EFFICIENCY

Our multi-modal event tracking and evolution framework is suitable for multimedia documents from various social media platforms, which can not only effectively capture their multi-modal topics, but also obtain the evolutionary trends of social events and generate effective event summary details over time. Our proposed mmETM model can exploit the multi-modal property of social event, which can effectively model social media documents including long text with related images and learn the correlations between textual and visual modalities to separate the visual-representative topics and non-visual-representative topics.

CHAPTER 4

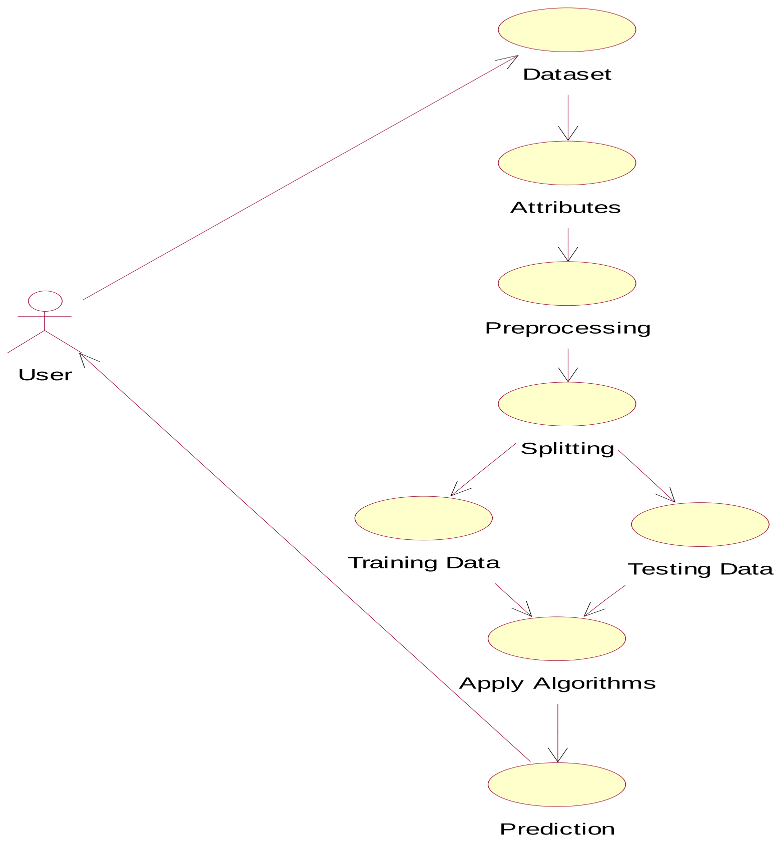
DESIGN ENGINEERING

4.1 GENERAL

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

UML Diagrams

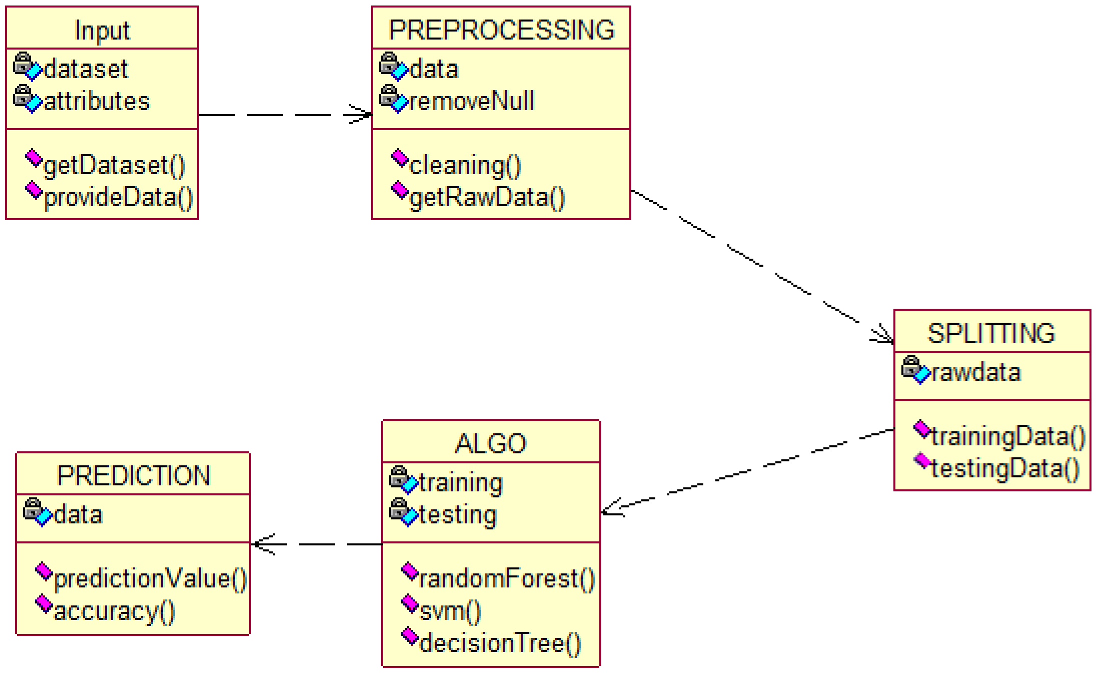
Use case diagram



EXPLANATION:

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

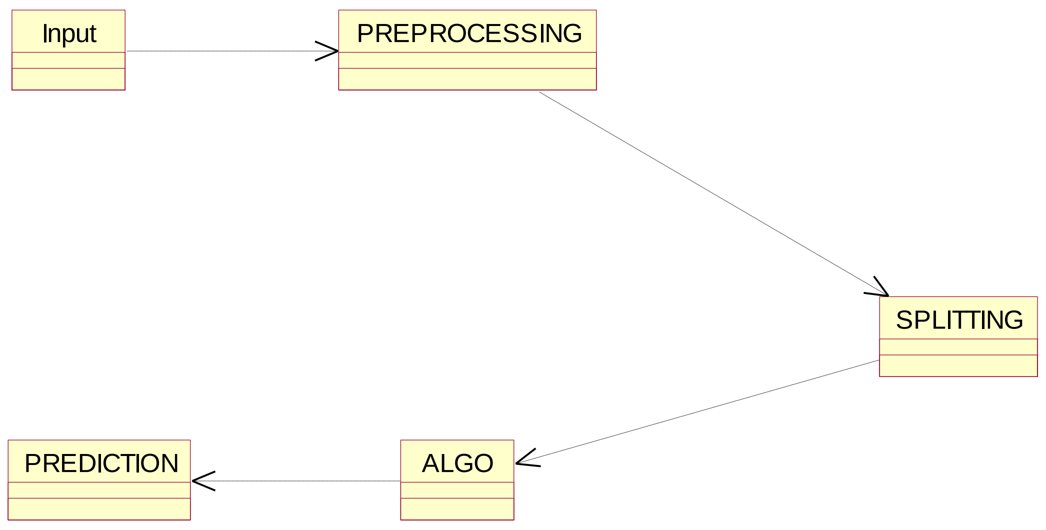
Class Diagram



EXPLANATION

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

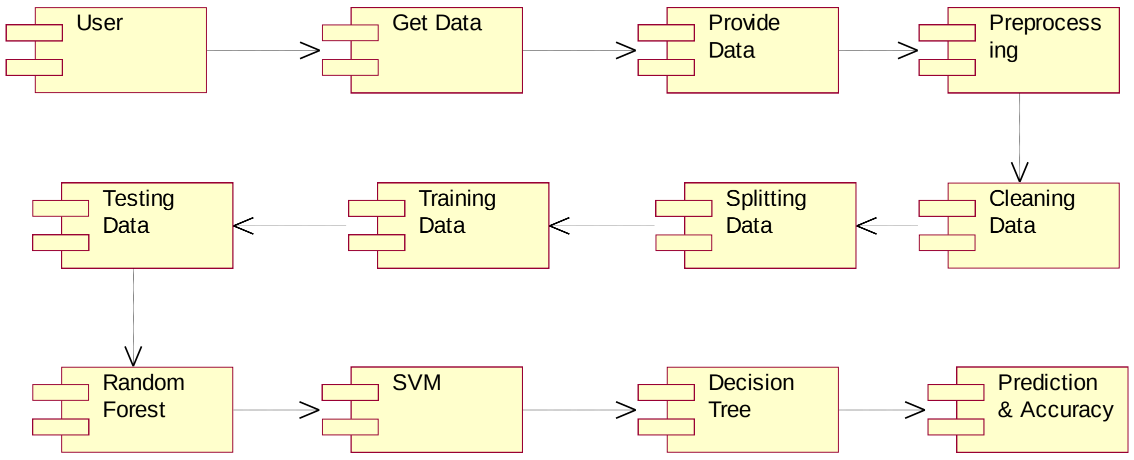
Object Diagram



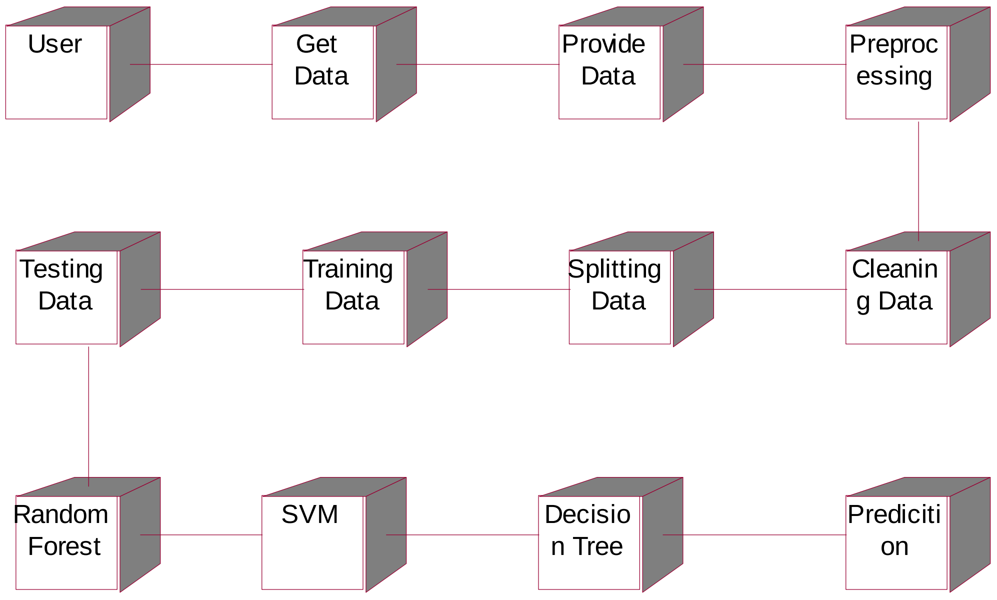
EXPLANATION:

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

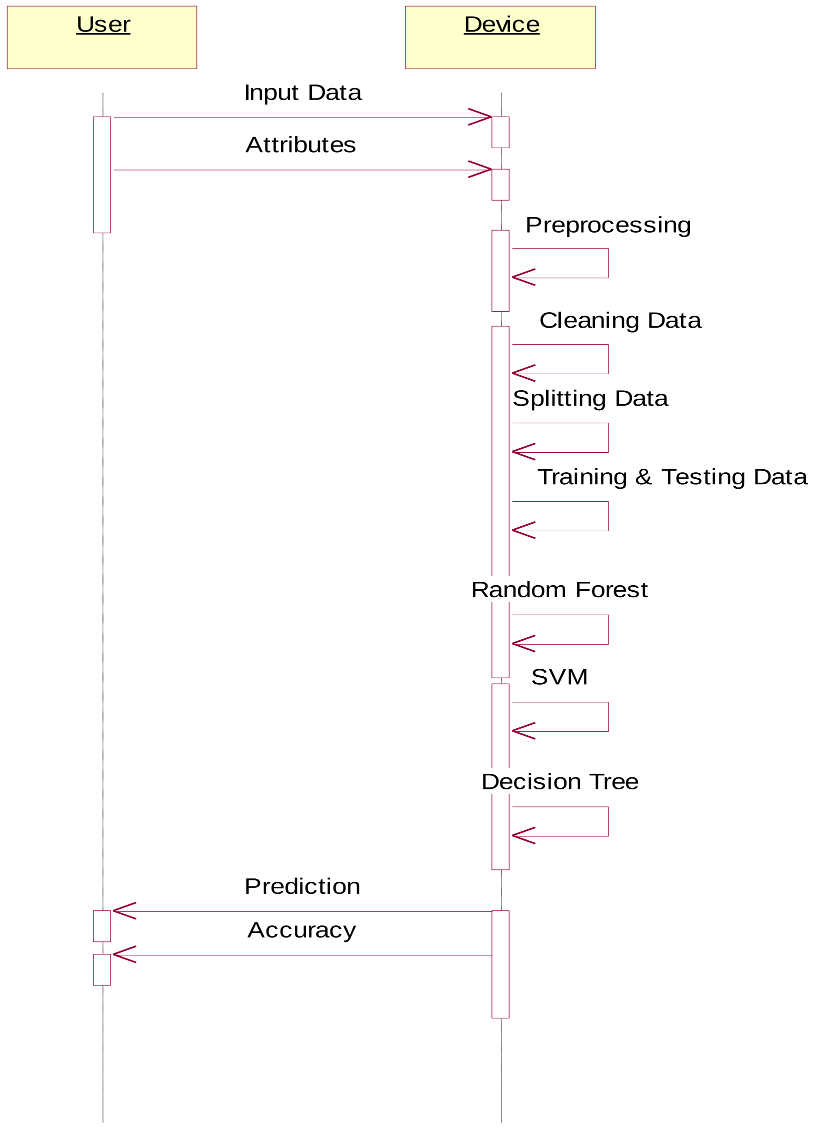
Component Diagram



Deployment Diagram



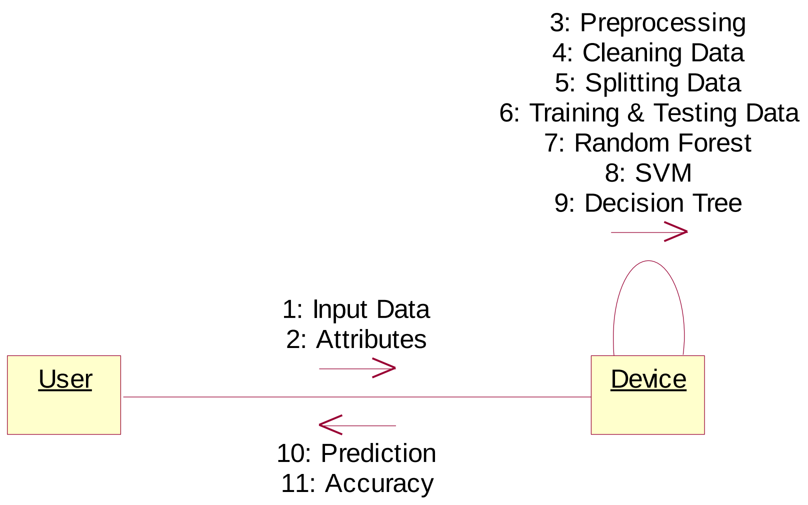
Sequence Diagram



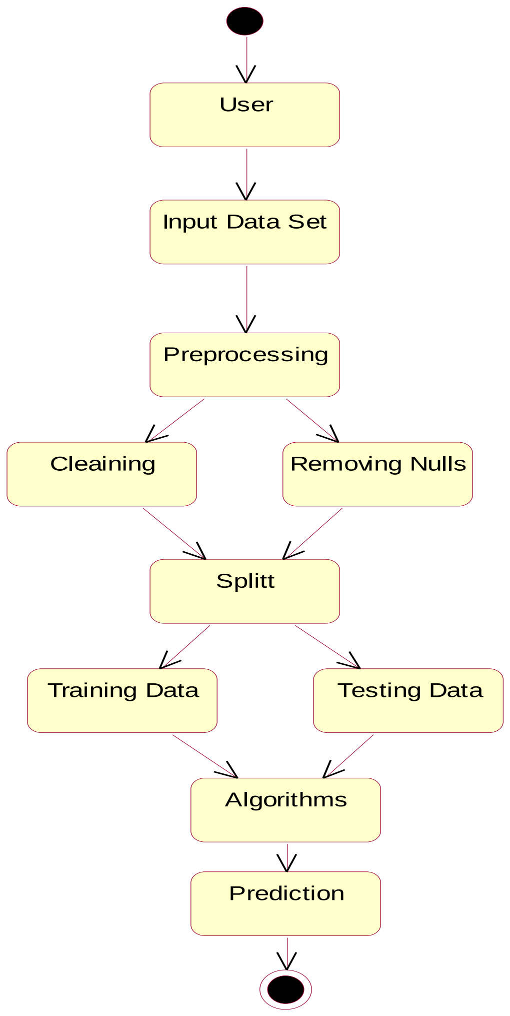
EXPLANATION:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

Collaboration Diagram



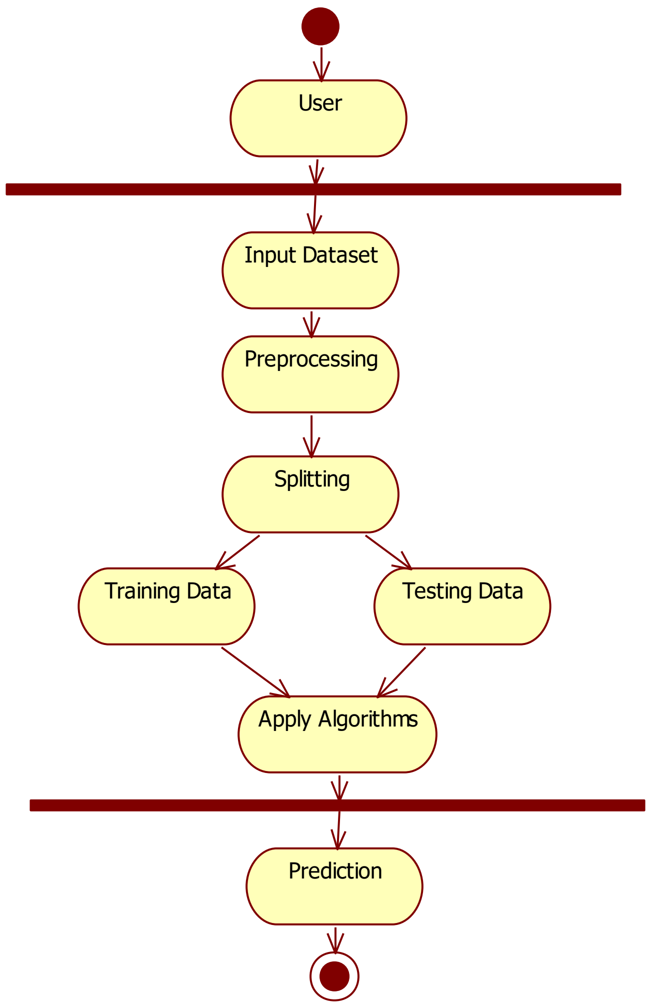
State Diagram



EXPLANATION:

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

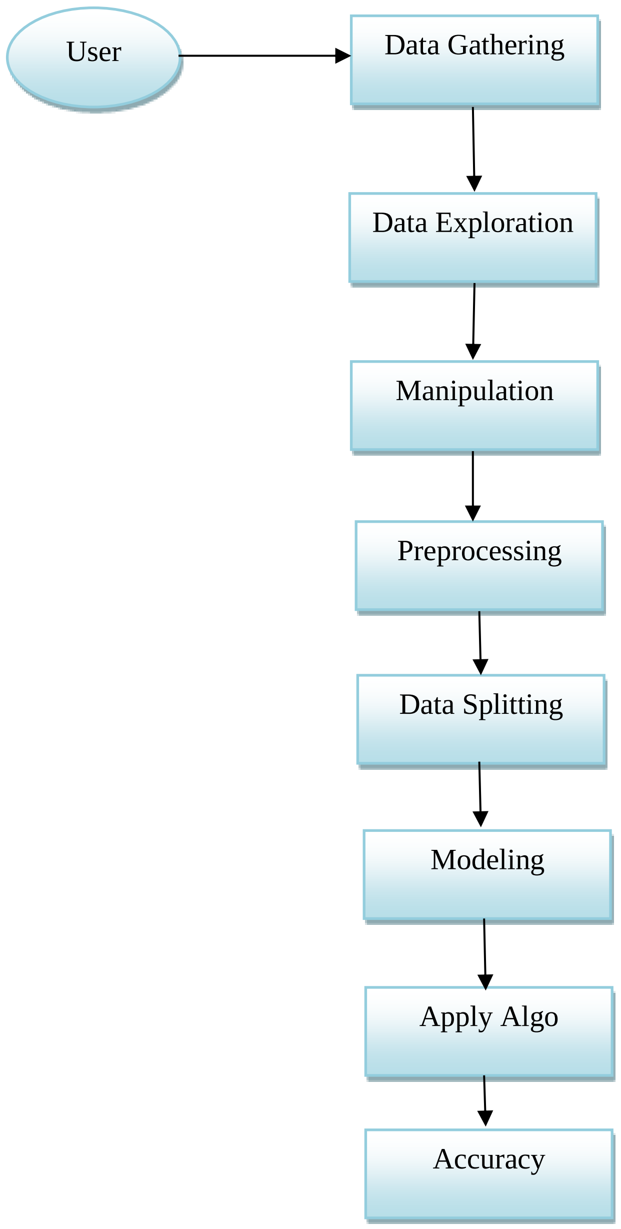
Activity Diagram



EXPLANATION:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

System Architecture



CHAPTER 5

DEVELOPMENT TOOLS

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.

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.

Importance of Python

- **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.

Features of Python

- **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.

Libraries used in python:

- numpy - mainly useful for its N-dimensional array objects.
- pandas - Python data analysis library, including structures such as dataframes.
- matplotlib - 2D plotting library producing publication quality figures.
- scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.

Figure : NumPy, Pandas, Matplotlib, Scikit-learn

CHAPTER 6

IMPLEMENTATION

6.1 GENERAL:

```
import cv2

import numpy as np

def get_output_layers(net):

        layer_names = net.getLayerNames()

       output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

        return output_layers

 def draw_prediction(img, class_id, confidence, x, y, x_plus_w, y_plus_h):

         label = str(classes[class_id])

        confidence=round(confidence*100)-1

        color = COLORS[class_id]

 cv2.rectangle(img, (x,y), (x_plus_w,y_plus_h), color, 2)

  cv2.putText(img, label+':'+str(confidence)+'%', (x-10,y-10), cv2.FONT_HERSHEY_SIMPLEX)

 image = cv2.imread('ped5.jpg')

 Width = image.shape[1]

  Height = image.shape[0]

  scale = 0.00392

  classes = None

 with open('yolov3.txt', 'r') as f:

 classes = [line.strip() for line in f.readlines()]

COLORS = np.random.uniform(0, 255, size=(len(classes), 3))

  net = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')

 blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), True, crop=False)

 net.setInput(blob)
```

```
outs = net.forward(get_output_layers(net))

class_ids = []

confidences = []

boxes = []

conf_threshold = 0.5

nms_threshold = 0.4

 for out in outs:

        for detection in out:

            scores = detection[5:]

            class_id = np.argmax(scores)

            confidence = scores[class_id]

           if confidence > 0.5:

              center_x = int(detection[0] * Width)

              center_y = int(detection[1] * Height)

              w = int(detection[2] * Width)

              h = int(detection[3] * Height)

              x = center_x - w / 2

              y = center_y - h / 2

              class_ids.append(class_id)

              confidences.append(float(confidence))

              boxes.append([x, y, w, h]

    indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)

    p_count=0

    for i in indices:

       i = i[0]

       box = boxes[i]
    x = box[0]

    y = box[1]

    w = box[2]

    h = box[3]

    if class_ids[i] == 0:

                p_count+=1

                draw_prediction(image, class_ids[i], confidences[i], round(x), round(y), round(x+w),
round(y+h))

         cv2.imshow("object detection", image)

        cv2.waitKey()

        cv2.imwrite("object-detection.jpg", image)

        cv2.destroyAllWindows()

        print("number of persons crossing pedestrains ",p_count)

        print(dtype('uint8'))#This means it is an 8ibit image (0-255)

        f.tofile('pedestrain.raw')

        import np

        fromraw=np.fromfile('pedestrain.raw',dtype=np.uint8)

        fromraw.shape

        fromraw.shape=(768, 1024, 3)

        memmap=np.memmap('pedestrain.raw',dtype=np.uint8,shape=(768,1024,3))

        f1=misc.face(gray=True) #For a grayscale image

        plt.imshow(f1,cmap=plt.cm.gray)

        plt.imshow(f1,cmap=plt.cm.gray,vmin=30,vmax=200)

                    # negative value means vehicle is moving UP

                    vehicle.vehicle_dir = -1

                  self.log.debug("Added match (%d, %d) to vehicle #%d. vector=(%0.2f,%0.2f)"
            , centroid[0], centroid[1], vehicle.id, vector[0], vector[1])
            return i
vehicle.frames_since_seen += 1
            self.log.debug("No match for vehicle #%d. frames_since_seen=%d"
            , vehicle.id, vehicle.frames_since_seen)
            return None
            def update_count(self, matches, output_image = None):
            self.log.debug("Updating count using %d matches...", len(matches))
            # First update all the existing vehicles
            for vehicle in self.vehicles:
            i = self.update_vehicle(vehicle, matches)
            if i is not None:
            del matches[i]
            #segmentation
            n,l=10,256
            im=np.zeros((l,l))
            np.random.seed(1)
            points=l*np.random.random((2,n**2))
            im[(points[0]).astype(np.int),(points[1]).astype(np.int)]=1
            im=ndimage.gaussian_filter(im,sigma=l/(4.*n))
            mask=(im>im.mean()).astype(np.float)
            mask+=0.1*im
            img=mask+0.2*np.random.randn(*mask.shape)
            hist,bin_edges=np.histogram(img,bins=60)
            bin_centers=0.5*(bin_edges[:-1]+bin_edges[1:])
```

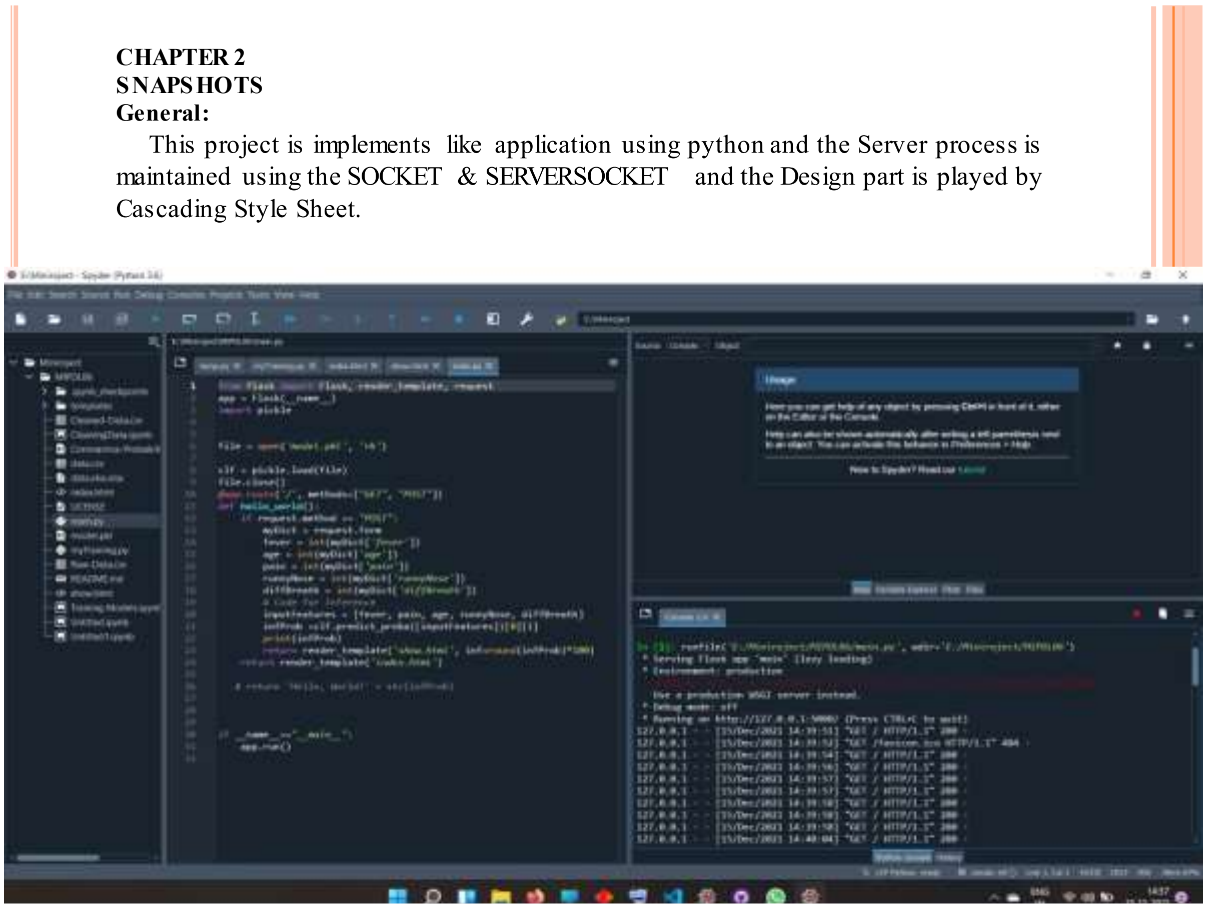
CHAPTER 7

SNAPSHOTS

General:

This project is implements like application using python and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

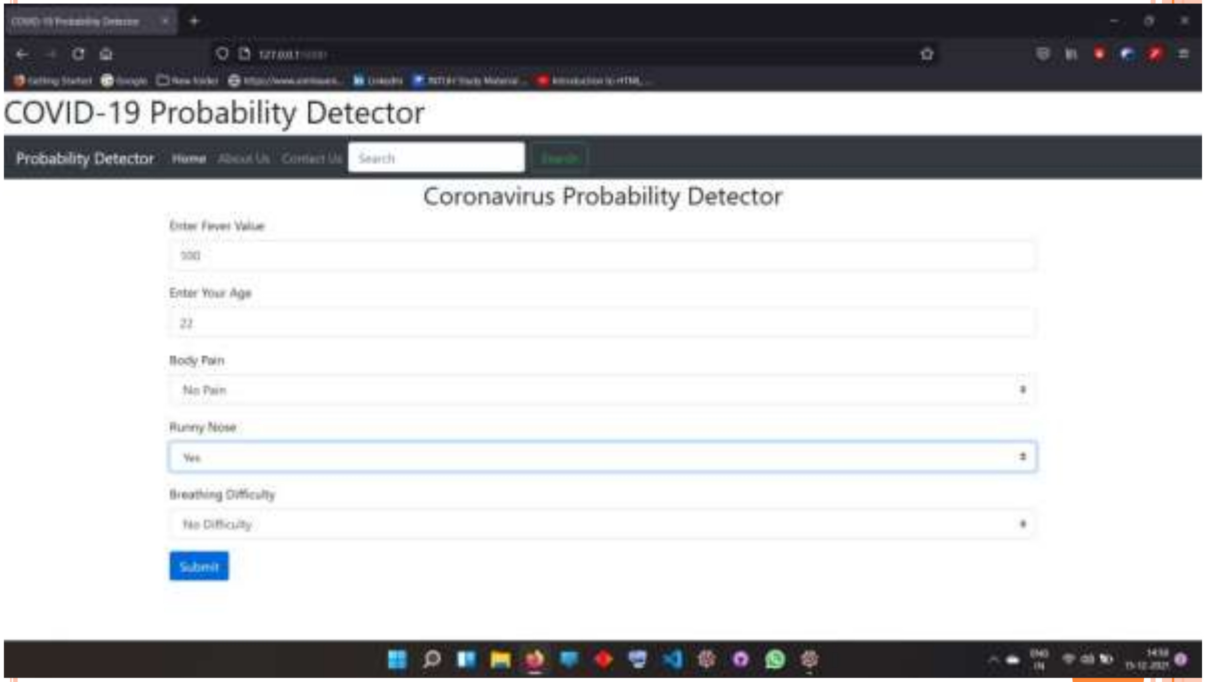
SNAPSHOTS



CHAPTER 2
SNAPSHOTS
General:
This project is implements like application using python and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

```
from flask import Flask, render_template, request
app = Flask(__name__)
import pickle


file = open('model.pkl', 'rb')

clf = pickle.load(file)
file.close()
@app.route('/', methods=["GET", "POST"])
def hello_world():
    if request.method == "POST":
        myDict = request.form
        fever = int(myDict['fever'])
        age = int(myDict['age'])
        pain = int(myDict['pain'])
        runnyNose = int(myDict['runnyNose'])
        diffBreath = int(myDict['diffBreath'])
        # Code for inference
        inputFeatures = [fever, pain, age, runnyNose, diffBreath]
        infProb =clf.predict_proba([inputFeatures])[0][1]
        print(infProb)
        return render_template('show.html', inf=round(infProb*100))
    return render_template('index.html')

    # return 'Hello, World!' + str(infProb)


if __name__=="__main__":
    app.run()
```

```
In [1]: runfile('E:/Miniproject/MYPOLO/main.py', wdir='E:/Miniproject/MYPOLO')
 * Serving Flask app 'main' (lazy loading)
 * Environment: production
   [illegible]
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [15/Dec/2021 14:39:51] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Dec/2021 14:39:52] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [15/Dec/2021 14:39:54] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Dec/2021 14:39:56] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Dec/2021 14:39:57] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Dec/2021 14:39:57] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Dec/2021 14:39:58] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Dec/2021 14:39:58] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Dec/2021 14:39:58] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Dec/2021 14:40:04] "GET / HTTP/1.1" 200 -
```


COVID-19 Probability Detector

Coronavirus Probability Detector

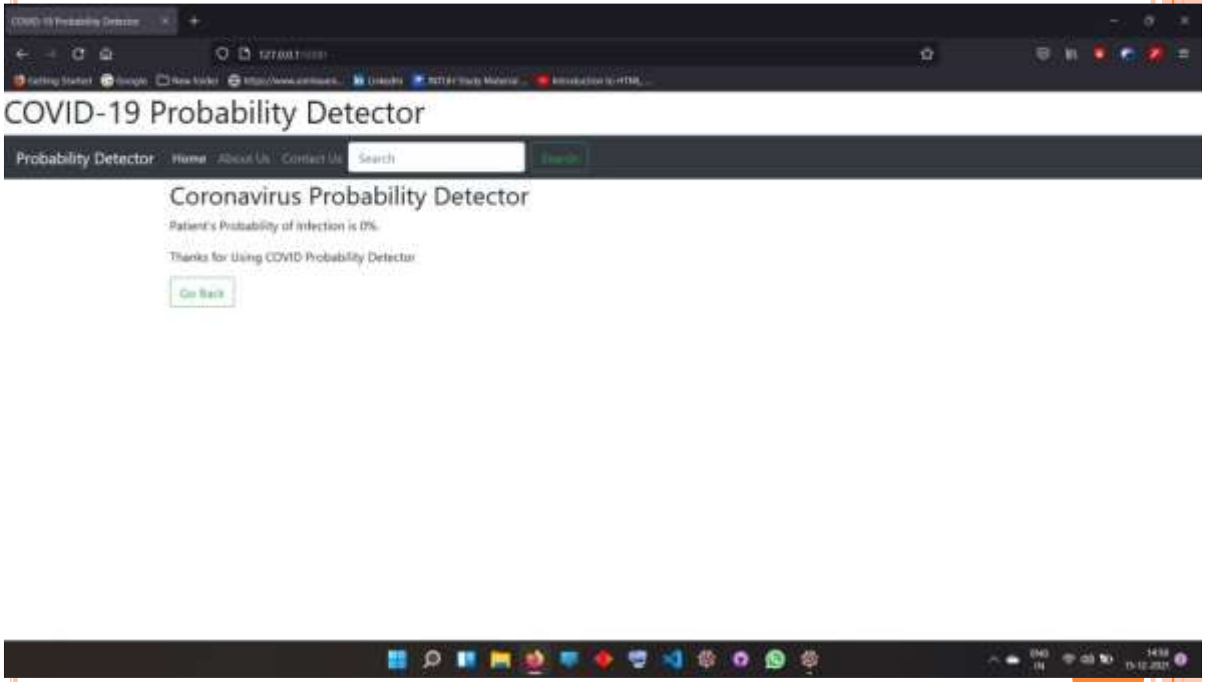
Enter Fever Value
100

Enter Your Age
22

Body Pain
No Pain

Runny Nose
Yes

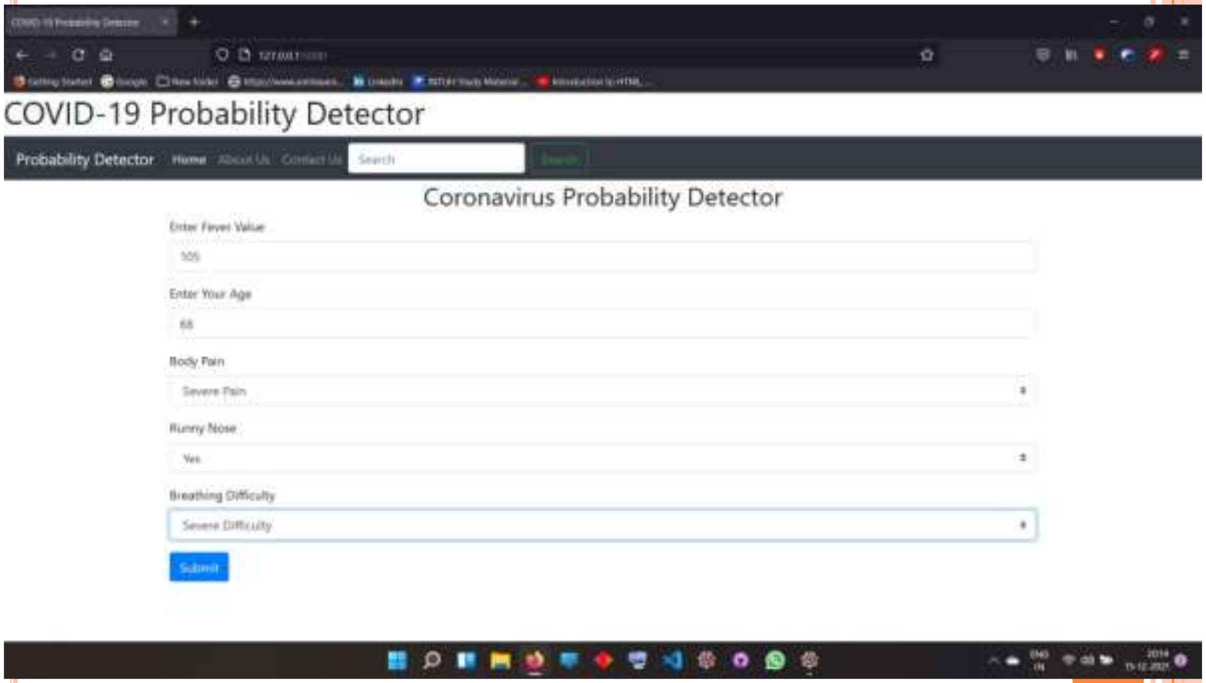
Breathing Difficulty
No Difficulty



Coronavirus Probability Detector

Patient's Probability of Infection is 0%.

Thanks for Using COVID Probability Detector



COVID-19 Probability Detector

Coronavirus Probability Detector

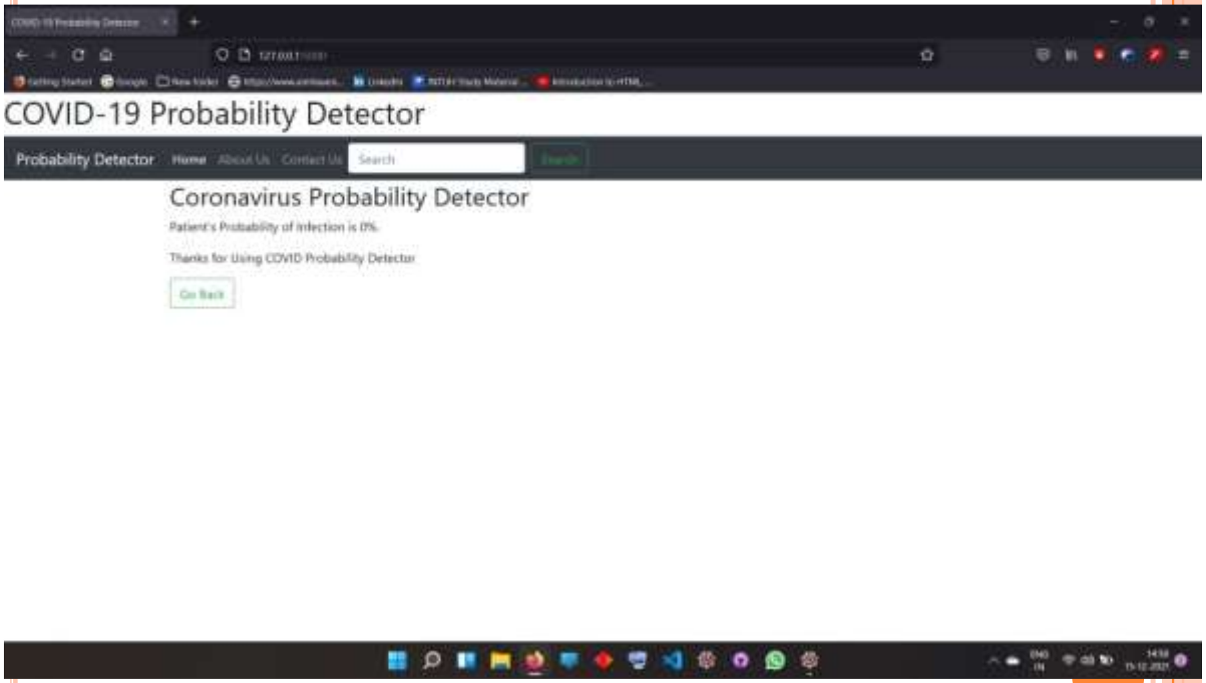
Enter Fever Value
105

Enter Your Age
68

Body Pain
Severe Pain

Runny Nose
Yes

Breathing Difficulty
Severe Difficulty


COVID-19 Probability Detector

Coronavirus Probability Detector

Patient's Probability of Infection is 0%.

Thanks for Using COVID Probability Detector

CHAPTER 8
SOFTWARE TESTING

8.1 GENERAL

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.

8.2 DEVELOPING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

8.3Types of Tests

8.3.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input 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.

8.3.2 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.

8.3.3 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.

8.3.4 Performance Test

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

8.3.5 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.

8.3.6 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.

Acceptance testing for Data Synchronization:

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updation process

8.2.7 Build the test plan

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

8.4 TEST CASES

Test scenario	Test steps	Test data	Expected result	Positive/Negative
1.Check whether covid is detected or not	Enter the symptoms	1.Enter fever value 2.Enter the age and body pain 3.Enter the breathing difficulty	Negative	Negative
1.Check whether covid is detected or not	Enter the symptoms	1.Enter fever value 2.Enter the age and body pain 3.Enter the breathing difficulty	Positive	Positive

CHAPTER 9

FUTURE ENHANCEMENT

Due to the complexity and cost issues involved in the process, such a device may prove beneficial when used for only large scale detection. The concept of sniffer for detecting lost mobiles paves a way to the recovery of lost cell phones. But still, this has not been practically implemented quite yet, so once the pricing and complexity of implementation can be brought down to reasonable levels, one can expect to see lot of practical implementations over the next few years.

CHAPTER 10

CONCLUSION & REFERENCE

9.1 CONCLUSION

The overall aim is to define various data mining techniques useful in effective COVID-19 disease prediction. Efficient and accurate prediction with a lesser number of attributes and tests is our goal. In this study, I consider only 14 essential attributes. I applied four data mining classification techniques, K-nearest neighbor, Naive Bayes, decision tree, and random forest. The data were pre-processed and then used in the model. K-nearest neighbor, Naïve Bayes, and random forest are the algorithms showing the best results in this model. I found the accuracy after implementing four algorithms to be highest in K-nearest neighbors (k = 7). We can further expand this research incorporating other data mining techniques such as time series, clustering and association rules, support vector machine, and genetic algorithm. Considering the limitations of this study, there is a need to implement more complex and combination of models to get higher accuracy for early prediction of COVID-19.

9.2 REFERENCES

1. Dong E, Du H, Gardner L. An interactive web-based dashboard to track COVID-19 in real time. The Lancet Infectious Diseases. Published online February 19, 2020. doi:10.1016/S1473-3099(20)30120-1

2. Gozes O, Frid-Adar M, Greenspan H, et al. Rapid AI Development Cycle for the Coronavirus (COVID-19) Pandemic: Initial Results for Automated Detection & Patient Monitoring using Deep Learning CT Image Analysis. arXiv e-prints. 2020;2003:arXiv:2003.05037. Accessed May 4, 2020. http://adsabs.harvard.edu/abs/2020arXiv200305037G

3. Song Y, Zheng S, Li L, et al. Deep learning Enables Accurate Diagnosis of Novel Coronavirus (COVID-19) with CT images. medRxiv. Published online February 25, 2020:2020.02.23.20026930. doi:10.1101/2020.02.23.20026930

4. Wang S, Kang B, Ma J, et al. A deep learning algorithm using CT images to screen for Corona Virus Disease (COVID-19). medRxiv. Published online April 24, 2020:2020.02.14.20023028. doi:10.1101/2020.02.14.20023028

5. Jin C, Chen W, Cao Y, et al. Development and Evaluation of an AI System for COVID-19 Diagnosis. medRxiv. Published online March 27, 2020:2020.03.20.20039834. doi:10.1101/2020.03.20.20039834

6. Tostmann A, Bradley J, Bousema T, et al. Strong associations and moderate predictive value of early symptoms for SARS-CoV-2 test positivity among healthcare workers, the Netherlands, March 2020. Eurosurveillance. 2020;25(16):2000508. doi:10.2807/1560-7917.ES.2020.25.16.2000508

7. Feng C, Huang Z, Wang L, et al. A Novel Triage Tool of Artificial Intelligence Assisted Diagnosis Aid System for Suspected COVID-19 pneumonia In Fever Clinics. medRxiv. Published online March 20, 2020:2020.03.19.20039099. doi:10.1101/2020.03.19.20039099

8. The Novel Coronavirus - Israel Ministry of Health. Accessed May 2, 2020. https://govextra.gov.il/ministry-ofhealth/corona/corona-virus-en/

9. COVID-19 - Government Data. Accessed May 2, 2020. https://data.gov.il/dataset/covid-19

10. Hastie T, Tibshirani R, Friedman J. Boosting and Additive Trees. In: Hastie T, Tibshirani R, Friedman J, eds. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. Springer; 2009:337-387. doi:10.1007/978-0-387-84858-7_10