

CHAPTER 1

INTRODUCTION

1.1 Introduction:

In general, for creating API the developers uses heavy weight communication services like REST API but here in this project the light weight communication protocols like MQTT, Web Sockets are used. In this project, a MQTT API is developed for all the micro-services which would be created by various developers working on a project. This project wants only the foreign developers (developers using other's code) to access our API instead of foreign code.

This API is created by using Flask and Fast API frameworks and these APIs run concurrently and do parallelism so that all developers can complete their given task. In this project, foreign developers should not exploit the foreign code (code written by others), and thus code synchronizing will be taken care of by this API.

This project is about creating a micro-services-based application where the developer would be deploying the tasks individually in AWS, and developer try to bind the web application and android communicate to the API.

1.2 Motivation:

Motivation behind this project is, for startups to create a server infrastructure, it would not be possible for them to maintain or manage servers in terms of financial and technical aspects. Thus, code will be shared among developers and code exploitation will be the scenario. To overcome that scenario, we are proposing a new tailored system where code synchronization will be taken care of us and containers will be running on our AWS server. Their job is to take data from our API and use it in their task.

1.3 Objective of Thesis

The main objective is to create a micro-services-based application where we would be deploying the tasks individually in our AWS, and we try to bind the web application to communicate with the API. This project wants only the foreign developers (developers using other's code) to access our API instead of foreign code.

1.4 Organization of Thesis

The rest of the thesis is organized in the following manner:

Chapter 2 -Deals with Literature Survey. Here some basic concepts and also protocols explained.

Chapter 3 - Contains the proposed work and analysis.

Chapter 4 -Gives the detailed description of Modules using UML Diagrams.

Chapter 5 -Gives the Implementation details, which the hardware and software requirements, description of the Technologies used.

Chapter 6 -Deals with the Debugging process of the proposed system.

Chapter 7 - Here the results of the implemented modules along with the respective screen shots are provided.

Chapter 8 - Here the Conclusions of the current application are provided.

Chapter 9 - Here the enhancement work for the future research are given.

Chapter 10 - Here the References, Text Books, Web Sites for this work are given.

Chapter 11- Here the publication details are given.

CHAPTER 2

LITERATURE SURVEY

2.1 Basic Concepts:

AWS (Amazon Web Services) is a comprehensive, evolving cloud computing platform provided by Amazon that includes a mixture of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offering.

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

In the “DEPLOYMENT OF PYTHON API ON AWS” project, we are creating a RESTful API for all the micro-services i.e., here in this project, we take healthcare-related micro-service which takes symptoms as input and gives which disease its related to and we want only the foreign developers (developers using other's code) to access our API instead of foreign code.

We would be creating an API by using Flask and FastAPI frameworks and these APIs run concurrently and do parallelism so that all developers can complete their given task. In this project, we don't want foreign developers to exploit the foreign code (code written by others), and thus code synchronizing will be taken care of our API.

2.2 Related Work:

For startups to create a server infrastructure, it would not be possible for them to maintain or manage servers in terms of financial and technical aspects. Thus, code will be shared among developers, and code exploitation will be the scenario.

- To overcome that scenario, we are proposing a new tailored system where code synchronization will be taken care of us and containers will be running on our AWS server. Their job is to take data from our API and use it in their task.



Figure 2.1: Block Diagram of Existing System

2.3 Drawbacks of existing system

For startups to create a server infrastructure, it would not be possible for them to maintain or manage servers in terms of financial and technical aspects. Thus, code will be shared among developers and code exploitation will be the scenario. Generally the foreign developers exploit the foreign code (code written by others). The main disadvantage of existing system is that, the user has to handle the API services provided by the developer. There is no support system between the user and API developer. The communication between the user and the API is a heavy weight communication.

CHAPTER 3

PROPOSED WORK & ANALYSIS

3.1 Introduction to API

API stands for Application Programming Interface. In the context of APIs, the word Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses. Their API documentation contains information on how developers are to structure those requests and responses.

APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols. For example, the weather bureau's software system contains daily weather data. The weather app on your phone "talks" to this system via APIs and shows you daily weather updates on your phone.

API architecture is usually explained in terms of client and server. The application sending the request is called the client, and the application sending the response is called the server. So in the weather example, the bureau's weather database is the server, and the mobile app is the client.

3.1.1 REST APIs

REST stands for Representational State Transfer. REST defines a set of functions like GET, PUT, DELETE, etc. that clients can use to access server data. Clients and servers exchange data using HTTP.

The main feature of REST API is statelessness. Statelessness means that servers do not save client data between requests. Client requests to the server are similar to URLs you type in your browser to visit a website. The response from the server is plain data, without the typical graphical rendering of a web page.

3.1.2 Benefits of REST APIs

REST APIs offer four main benefits:

1. Integration

APIs are used to integrate new applications with existing software systems. This increases development speed because each functionality doesn't have to be written from scratch. You can use APIs to leverage existing code.

2. Innovation

Entire industries can change with the arrival of a new app. Businesses need to respond quickly and support the rapid deployment of innovative services. They can do this by making changes at the API level without having to re-write the whole code.

3. Expansion

APIs present a unique opportunity for businesses to meet their clients' needs across different platforms. For example, maps API allows map information integration via websites, Android, iOS, etc. Any business can give similar access to their internal databases by using free or paid APIs.

4. Ease of maintenance

The API acts as a gateway between two systems. Each system is obliged to make internal changes so that the API is not impacted. This way, any future code changes by one party do not impact the other party.

3.1.3 Flask framework

Flask is a widely used micro web framework for creating APIs in Python. It is a simple yet powerful web framework which is designed to get started quick and easy, with the ability to scale up to complex applications.

3.2 Amazon Web Services:

3.2.1 Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

3.2.2 Features of Amazon EC2

Amazon EC2 provides the following features:

- Virtual computing environments, known as instances
- Preconfigured templates for your instances, known as Amazon Machine Images (AMIs), that package the bits you need for your server (including the operating system and additional software)
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types
- Secure login information for your instances using key pairs (AWS stores the public key, and you store the private key in a secure place)
- Storage volumes for temporary data that's deleted when you stop, hibernate, or terminate your instance, known as instance store volumes

- Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes
- Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as Regions and Availability Zones
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups
- Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses
- Metadata, known as tags, that you can create and assign to your Amazon EC2 resources
- Virtual networks you can create that are logically isolated from the rest of the AWS Cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs)

3.3 ML model:

In this project we are developing a ml model using the API, this model identifies the diseases based on the symptoms provided by the user.

The model is developed using decision tree classifier algorithm. After developing the model, we have to train the model with a dataset not only training the model we have to test the model.

3.3.1 Decision Tree Classification

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes

represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- Below diagram explains the general structure of a decision tree:
- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

3.4 DEPLOYMENT OF PYTHON API ON AWS

Deployment involves packaging up your web application and putting it in a production environment that can run the app. Deployment of python API on AWS includes several steps.

Step 1: create an EC2 instance

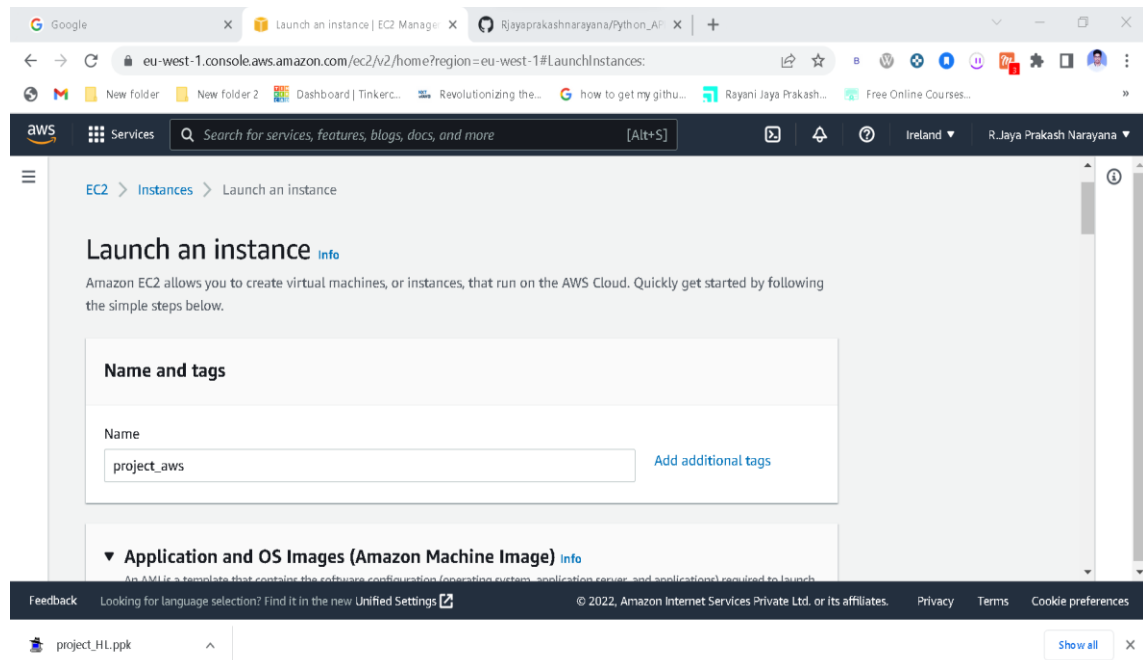


Fig 3.1: Launching an EC2 instance to deploy API

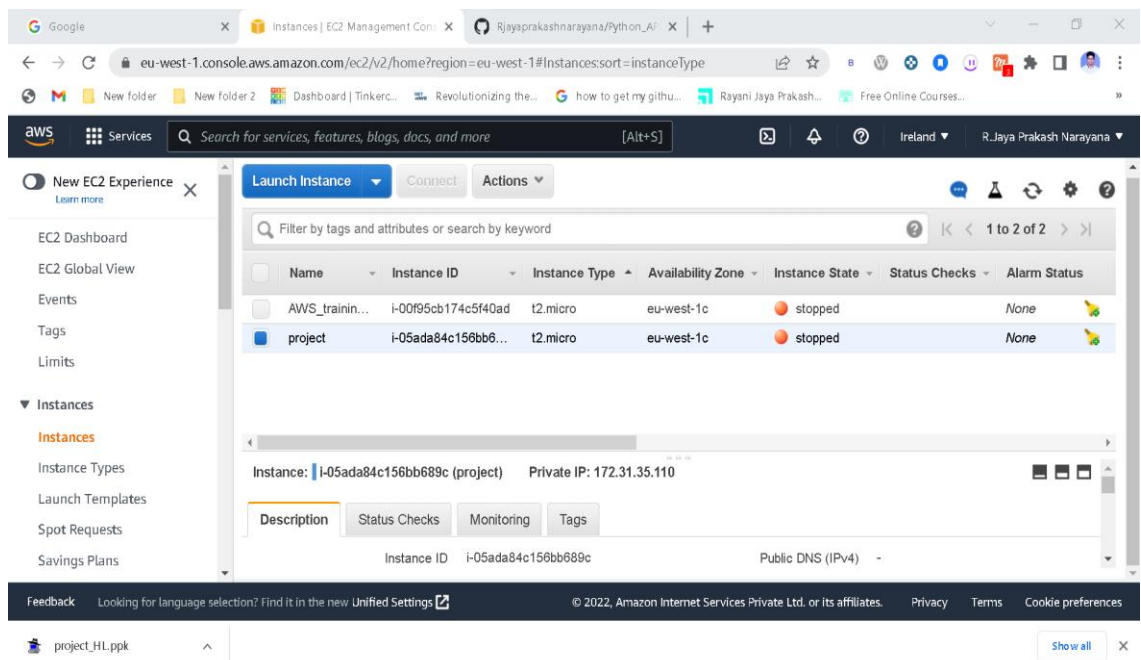
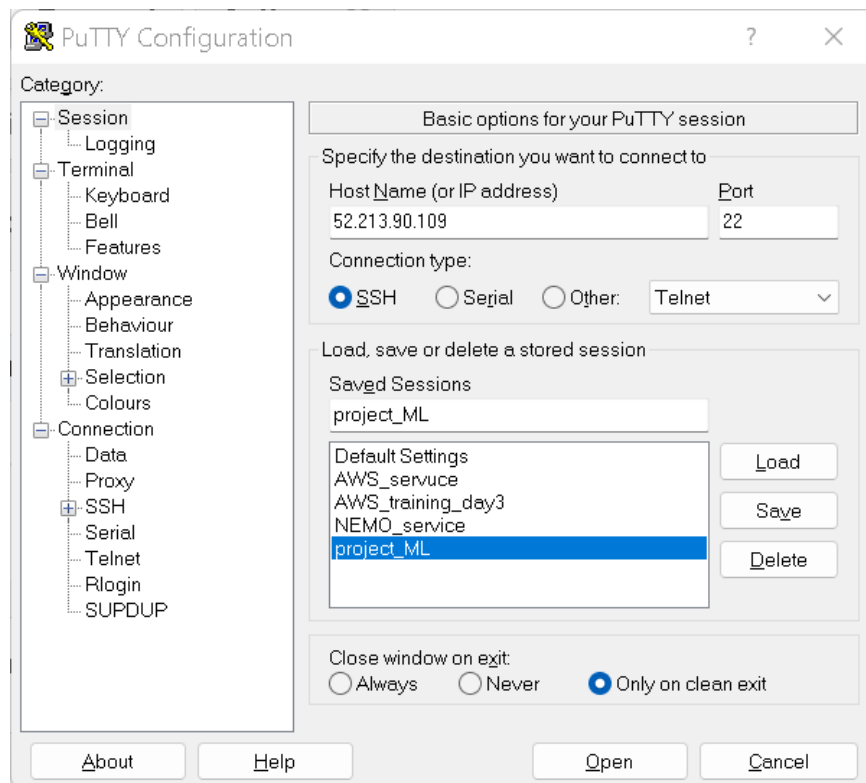
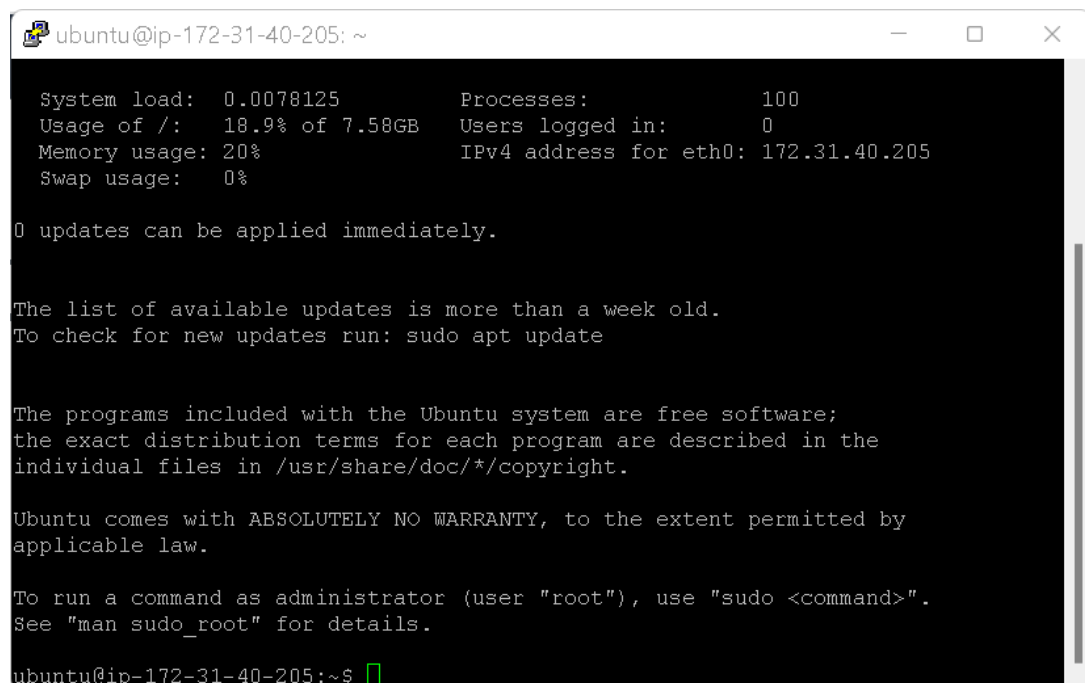
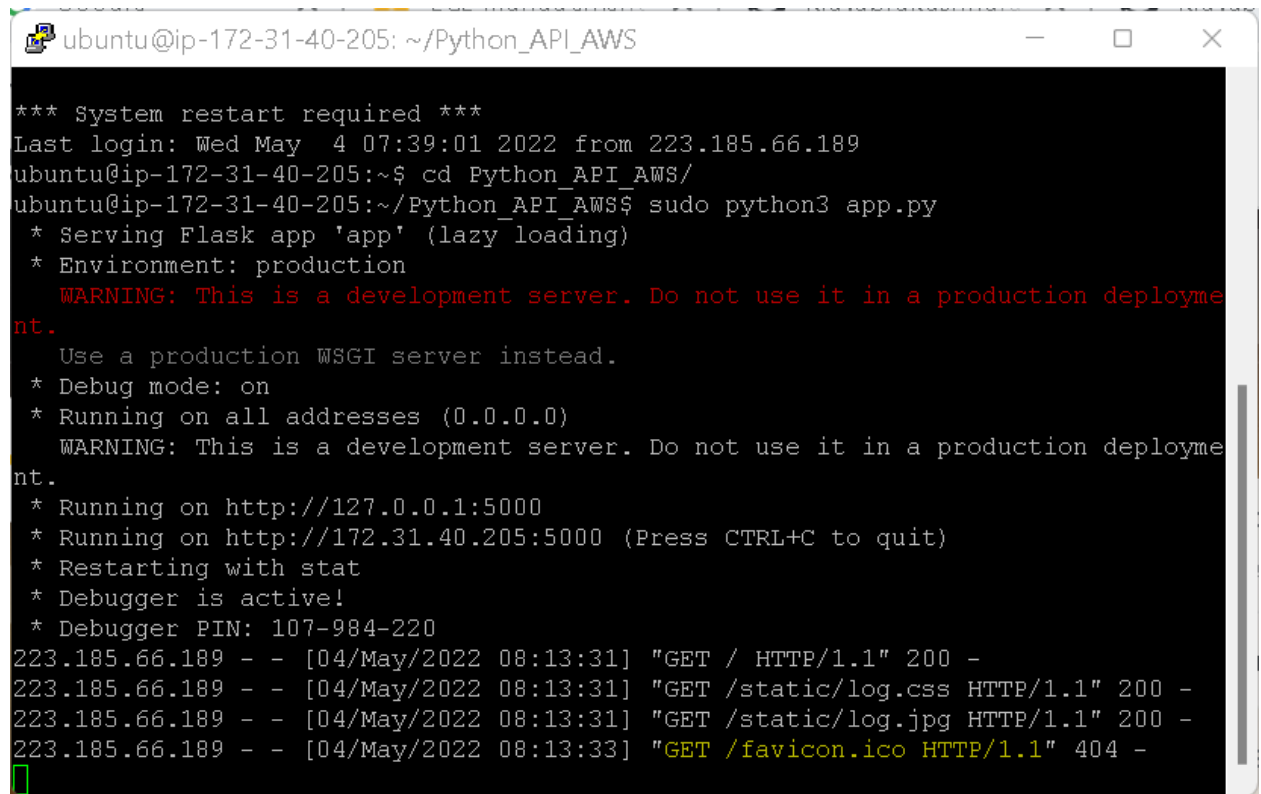


Fig 3.2 EC2 Instance

Step 2: Launching putty to connect the server

**Fig 3.3: configuring putty settings****Fig 3.4: Launching of ubuntu server**

Step 3: Deployment of Python API on EC2 instance



```
ubuntu@ip-172-31-40-205: ~/Python_API_AWS

*** System restart required ***
Last login: Wed May  4 07:39:01 2022 from 223.185.66.189
ubuntu@ip-172-31-40-205:~$ cd Python_API_AWS/
ubuntu@ip-172-31-40-205:~/Python_API_AWS$ sudo python3 app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.40.205:5000 (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 107-984-220
223.185.66.189 - - [04/May/2022 08:13:31] "GET / HTTP/1.1" 200 -
223.185.66.189 - - [04/May/2022 08:13:31] "GET /static/log.css HTTP/1.1" 200 -
223.185.66.189 - - [04/May/2022 08:13:31] "GET /static/log.jpg HTTP/1.1" 200 -
223.185.66.189 - - [04/May/2022 08:13:33] "GET /favicon.ico HTTP/1.1" 404 -
```

Fig 3.5: Deployment of Python API on EC2

CHAPTER 4

DESIGN

4.1 System Design

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system.

The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities - design, code and test that required building and verifying software.

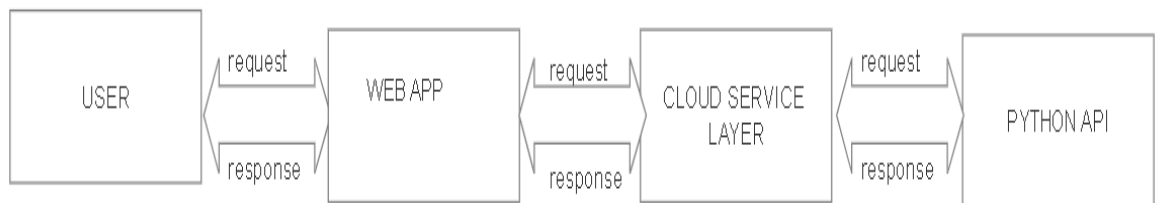


figure 4.1 Block diagram of proposed system

4.2 UML DIAGRAMS

4.2.1 UML Data

The UML's four structural diagram exists to visualize, specify, construct and document the static aspects of system. Just as the static aspects of a house encompass the existence and placement of such things as walls, doors, windows, pipes, wires and vents. So to do the static aspects of a software system

encompass the existence and placement of such things classes, interfaces, collaborations, components and nodes.

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the art effects of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

- actors
- business processes
- (logical) components
- activities
- programming language statements
- database schemas, and
- Reusable software components.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies.

UML has synthesized the notations of the Booth method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

4.2.2 Use case diagram

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. UML use case diagrams are ideal for:

- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system
- Modelling the basic flow of events in a use case

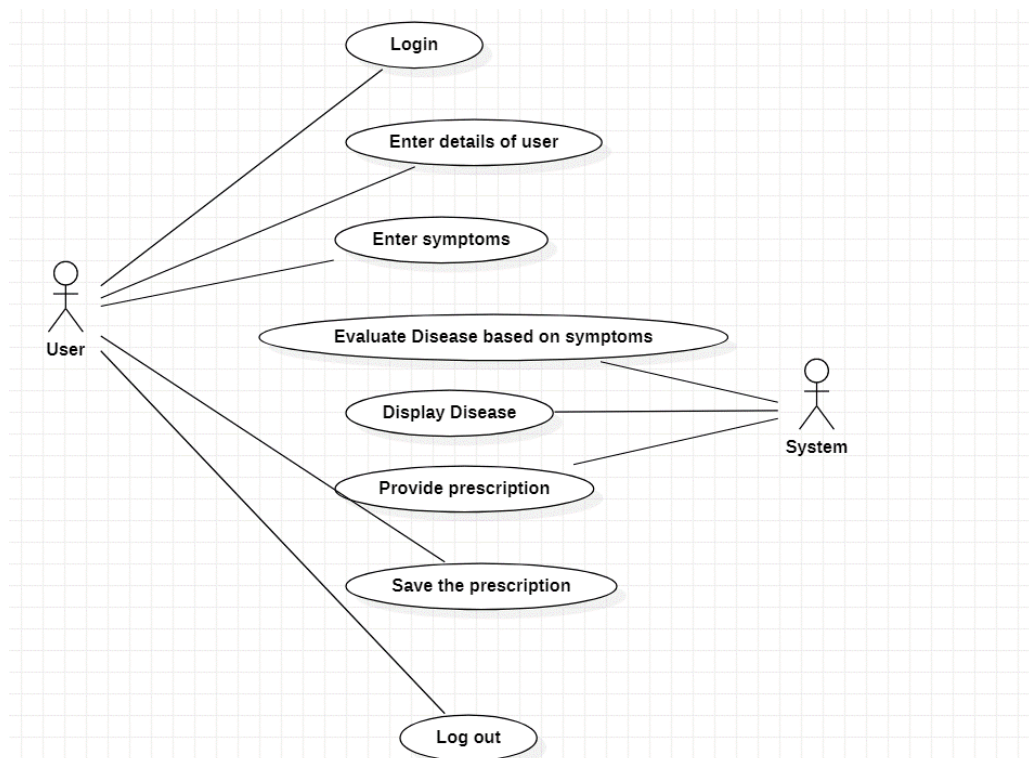


Figure 4.2 Use case diagram

4.2.3 Class Diagram

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

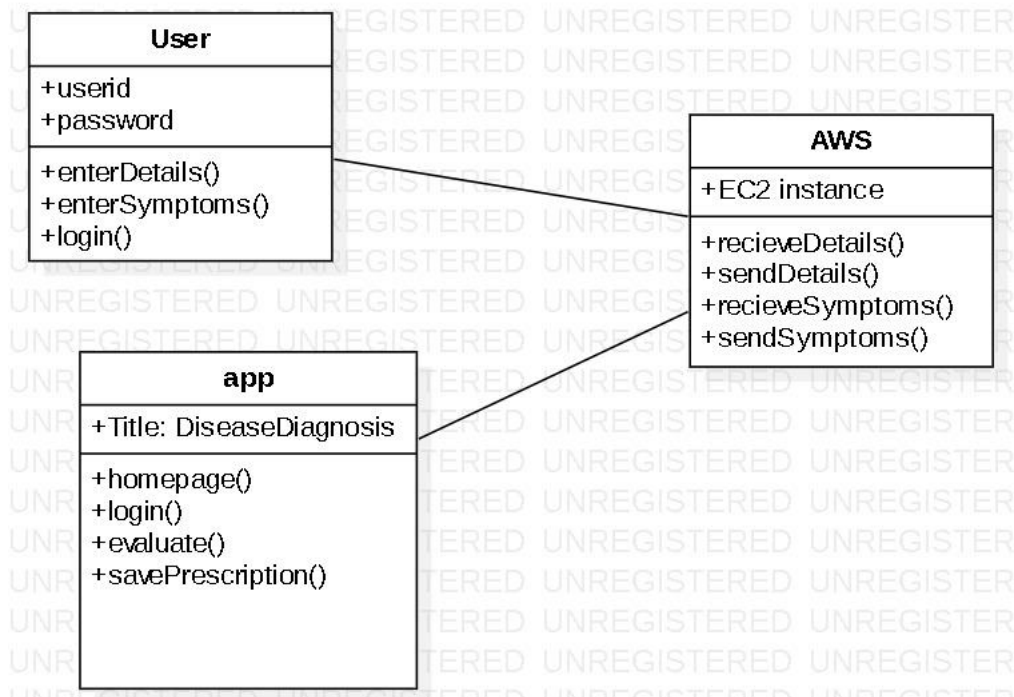


Figure 4.3 Class Diagram

4.2.4 Sequence Diagram

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

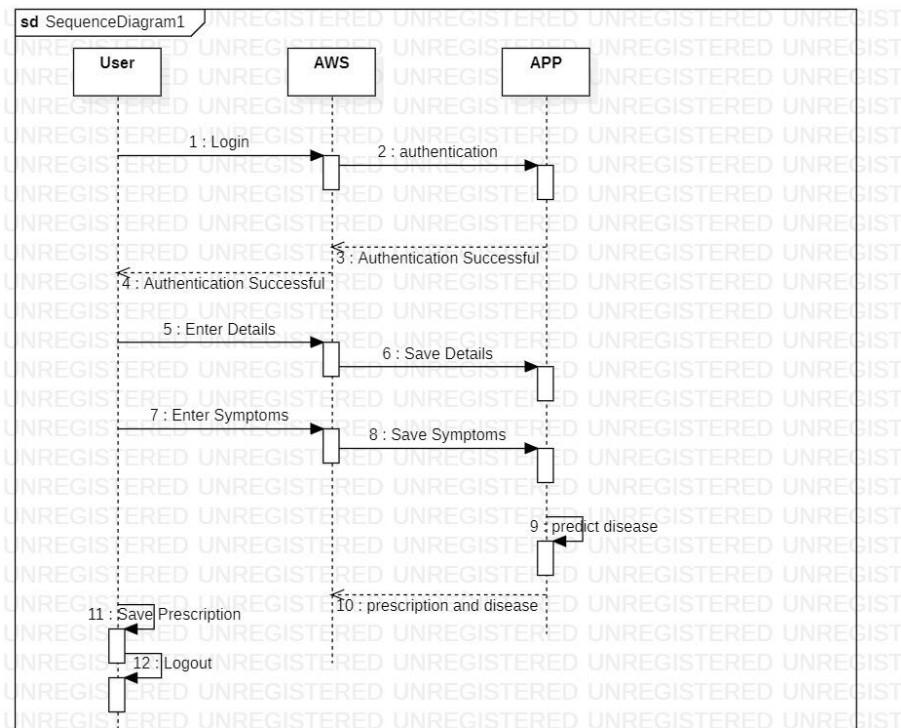


Figure 4.4 Sequence Diagram

4.2.5 Deployment Diagram

A deployment diagram models the run-time architecture of a system. It shows the configuration of the hardware elements (Nodes) and shows how software elements and artifacts are mapped onto these nodes. A Node is either a hardware or software element.

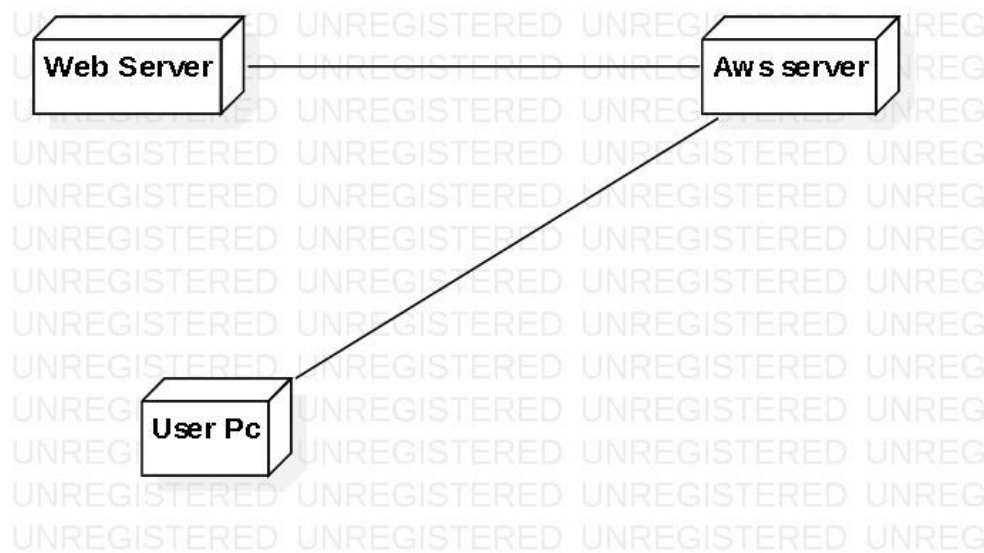


Figure 4.5: Deployment Diagram

CHAPTER 5

IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

5.1 Software Requirements

- Python
- AWS EC2
- VS code

5.2 Hardware Requirements

- RAM - 16GB
- Secondary Storage Space – 25GB
- I3 and above processor

5.3 Technologies Used

5.3.1 Python Versions:

- Python 2.0 was released on 16 October 2000 and had many major new features, including a cycle-detecting, garbage collector, and support for Unicode. With this release, the development process became more transparent and community-backed.

- Python 3.0 (initially called Python 3000 or py3k) was released on 3 December 2008 after a long testing period. It is a major revision of the language that is not completely backward-compatible with previous versions. However, many of its major features have been back ported to the Python 2.6.x and 2.7.x version series, and releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.
- Python 2.7's end-of-life date (a.k.a. EOL, sunset date) was initially set at 2015, then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. In January 2017, Google announced work on a Python 2.7 to go Trans compiler to improve performance under concurrent workloads.
- Python 3.6 had changes regarding UTF-8 (in Windows, PEP 528 and PEP 529) and Python 3.7.0b1 ([PEP 540](#)) adds a new "UTF-8 Mode" (and overrides [POSIX locale](#)).

5.3.2 Python IDLE

- **IDLE** is Python's **Integrated Development and Learning Environment**. It allows programmers to easily write Python code. Just like Python Shell, IDLE can be used to execute a single statement and create, modify, and execute Python scripts.
- IDLE provides a fully-featured text editor to create Python scripts that include features like syntax highlighting, autocompletion, and smart indent. It also has a debugger with stepping and breakpoints features. This makes debugging easier.
- **IDLE** is Python's **Integrated Development and Learning Environment**. It allows programmers to easily write Python code. Just like Python Shell, IDLE can

be used to execute a single statement and create, modify, and execute Python scripts.

- IDLE provides a fully-featured text editor to create Python scripts that include features like syntax highlighting, auto completion, and smart indent. It also has a debugger with stepping and breakpoints features. This makes debugging easier.

In IDLE, we write code line by line. Each specific line will handle one thing, and you can type whatever you want in that line and press enter to execute it. IDLE works more like a terminal or command prompt - You write one line, press enter, it executes.

5.4 Modules:

List of Modules

List of Modules:

- API module
- AWS System module

5.4.1 API module:

5.4.1.1 Introduction to API

API stands for Application Programming Interface. In the context of APIs, the word Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses. Their API documentation contains information on how developers are to structure those requests and responses.

APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols. For example, the weather bureau's software system contains daily weather data. The weather app on your phone "talks" to this system via APIs and shows you daily weather updates on your phone.

API architecture is usually explained in terms of client and server. The application sending the request is called the client, and the application sending the response is called the server. So in the weather example, the bureau's weather database is the server, and the mobile app is the client.

5.4.1.2 REST APIs

REST stands for Representational State Transfer. REST defines a set of functions like GET, PUT, DELETE, etc. that clients can use to access server data. Clients and servers exchange data using HTTP.

The main feature of REST API is statelessness. Statelessness means that servers do not save client data between requests. Client requests to the server are similar to URLs you type in your browser to visit a website. The response from the server is plain data, without the typical graphical rendering of a web page.

5.4.1.3 Benefits of REST APIs

REST APIs offer four main benefits:

1. Integration

APIs are used to integrate new applications with existing software systems. This increases development speed because each functionality doesn't have to be written from scratch. You can use APIs to leverage existing code.

2. Innovation

Entire industries can change with the arrival of a new app. Businesses need to respond quickly and support the rapid deployment of innovative services. They can do this by making changes at the API level without having to re-write the whole code.

3. Expansion

APIs present a unique opportunity for businesses to meet their clients' needs across different platforms. For example, maps API allows map information integration via websites, Android, iOS, etc. Any business can give similar access to their internal databases by using free or paid APIs.

4. Ease of maintenance

The API acts as a gateway between two systems. Each system is obliged to make internal changes so that the API is not impacted. This way, any future code changes by one party do not impact the other party.

5.4.2 AWS System module:

5.4.2.1 Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

5.4.2.2 Features of Amazon EC2

Amazon EC2 provides the following features:

- Virtual computing environments, known as instances

- Preconfigured templates for your instances, known as Amazon Machine Images (AMIs), that package the bits you need for your server (including the operating system and additional software)
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types
- Secure login information for your instances using key pairs (AWS stores the public key, and you store the private key in a secure place)
- Storage volumes for temporary data that's deleted when you stop, hibernate, or terminate your instance, known as instance store volumes
- Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes
- Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as Regions and Availability Zones
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups
- Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses
- Metadata, known as tags, that you can create and assign to your Amazon EC2 resources
- Virtual networks you can create that are logically isolated from the rest of the AWS Cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs)

5.5 ALGORITHM USED:

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- Below diagram explains the general structure of a decision tree:

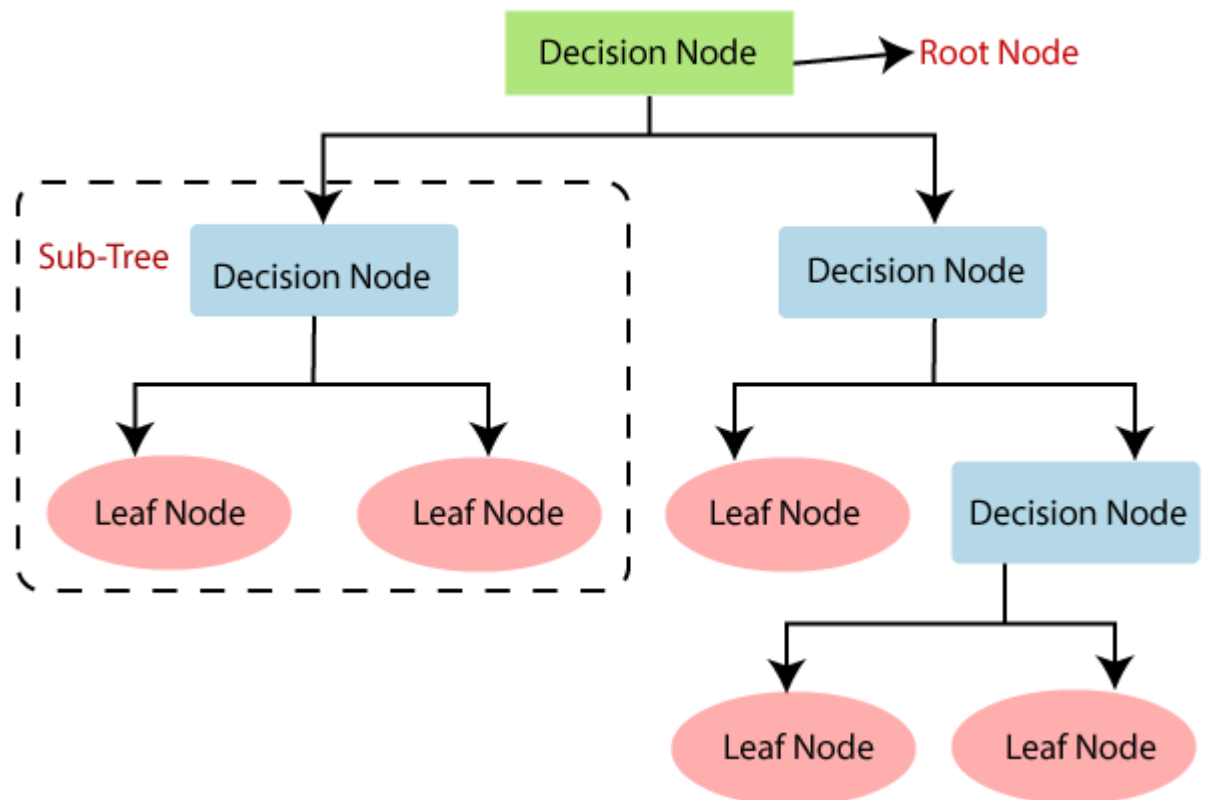


Figure 5.1: Structure of a decision tree

- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.
- Decision Tree Classification Algorithm involves the following steps:
 - Step 1:** Begin the tree with the root node, says S, which contains the complete dataset.
 - Step 2:** Find the best attribute in the dataset using Attribute Selection Measure (ASM).
 - Step 3:** Divide the S into subsets that contains possible values for the best attributes.
 - Step 4:** Generate the decision tree node, which contains the best attribute.

Step 5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

5.6 Coding

5.6.1 API Code

```
import pickle

from flask import Flask,render_template,request

import pandas as pd

data = pd.read_csv('Training.csv')

X = data.iloc[:, :-1]

l1=X.columns

l2=[]

for i in range(0,len(l1)):

    l2.append(0)

model = pickle.load(open('model.pkl','rb'))

app = Flask(__name__)

@app.route('/')

def homepage():

    return render_template('log.html')

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

def login():
```

```
username = request.form['userid']

password = request.form['pwd']

if username=='demo' and password=='1234':

    return render_template('reg.html')

else:

    return render_template('log.html',err_log='invalidcredentials')


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

def evaluate():

    name = request.form['name']

    aadhaar = request.form['aadhaar']

    district = request.form['district']

    village = request.form['village']

    phone =request.form['phone']

    age = request.form['age']

    gender = request.form['gender']

    symptoms =request.form['symptoms']

    symptoms = symptoms.split("\r\n")

    for j in range(0,len(l1)):

        for k in symptoms:

            if k==l1[j]:

                l2[j]=1

    inputtest =[l2]

    out = model.predict(inputtest)

    h = out[0]
```

```
for j in range(0,len(11)):

    l2[j]=0

    if (h=='Fungal infection'):

        medicines='Fluconazole tab , if chronic - voriconazole tab (Fungal
Infection)'

    elif(h=='Allergy'):

        medicines='Cetirizine tab , Levocetirizine tab , Loratidine tab , '

    elif(h=='GERD'):

        medicines='Raveprazole tab , Lansoprazole tab , Esomeprazole tab ,
Dexlansoprazole tab'

    elif(h=='Chronic cholestasis'):

        medicines='Piparcillin tazobactum injection'

    elif(h=='Drug Reaction'):

        medicines='Avil tab'

    elif(h=='Peptic ulcer diseae'):

        medicines='Raveprazole tab , Lansoprazole tab , Esomeprazole tab ,
Dexlansoprazole tab'

    elif(h=='AIDS'):

        medicines='Dolutegravir tab , Antiretroviral drug , Tenofovar tab ,
Lamivudine tab , Triomune tab , Darunavir tab'

    elif(h=='Diabetes'):

        medicines='Glimepride tab , Metformin tab , Gliclazide tab'

    elif(h=='Gastroenteritis'):

        medicines='Raveprazole tab \r, Lansoprazole tab \r, Esomeprazole tab \r,
Dexlansoprazole tab'
```

```
elif(h=='Bronchial Asthma'):

    medicines='Doxifillin tab , Levosolbutamol Bromexyne syrup ,
Asebrofithillin tab , Theophylline tab , Buducorte inhaler , Foracorte Forte
inhaler'

elif(h=='Hypertension'):

    medicines='Amlodipin tab , Diltiazem tab , Isravipine tab , Nicardipine
tab , Verapimil tab'

elif(h=='Migraine'):

    medicines='Paracetamol tab , Naproxen tab'

elif(h=='Cervical spondylosis'):

    medicines='Indomethacin tab , Piroxicam tab , Naproxen tab ,
Methylpregnisolin tab , Ibuprofen tab , Baclofen tab , Cyclobenzaprine tab ,
Aceclophenac'

elif(h=='Paralysis (brain hemorrhage)'):

    medicines='Anafranil tab , Clomipranime tab , methylphenidate tab'

elif(h=='Jaundice'):

    medicines='Live52 tab'

elif(h=='Malaria'):

    medicines='Ivermectin tab , Chloroquine Phosphate tab , Larinate 200mg
tab , injections 60mg , Mefloquine Artesunate tab'

elif(h=='Chicken pox'):

    medicines='Immune System , Acyclovir tab 800mg , Valacyclovir tab'

elif(h=='Dengue'):

    medicines='Paracetamol , Aspirin , Immune System'

elif(h=='Typhoid'):
```

```
medicines='Ciprofloxacin tab , Ofloxacin tab , Norflaxacin tab ,  
Amoxycillin Clavu 625mg tab , Levofloxacin tab'  
  
elif(h=='hepatitis A'):  
  
    medicines='Get Doctor Prescription'  
  
elif(h=='Hepatitis B'):  
  
    medicines='Get Doctor Prescription'  
  
elif(h=='Hepatitis C'):  
  
    medicines='Get Doctor Prescription'  
  
elif(h=='Hepatitis D'):  
  
    medicines='Get Doctor Prescription'  
  
elif(h=='Hepatitis E'):  
  
    medicines='Get Doctor Prescription'  
  
elif(h=='Alcoholic hepatitis'):  
  
    medicines='Metadoxine tab'  
  
elif(h=='Tuberculosis'):  
  
    medicines='Ethambutol tab , Pyrazinamide tab , Cycloserine tab ,  
Ethionamide tab , Levofloxacin tab , Kanamycin tab'  
  
elif(h=='Common Cold'):  
  
    medicines='Immune System , Phenylephrine Hydrochloride +  
Chlorpheniramine Maleate tab , Levocetirizine tab , Cetirizine tab'  
  
elif(h=='Pneumonia'):  
  
    medicines='Azithromycin tab , Linezolid tab , Delafloxacin tab  
Clindamycin tab , Doxycycline tab , Ertapenem tab , Lincosamide tab'  
  
elif(h=='Dimorphic hemorrhoids(piles)'):  
  
    medicines='Pilex tab , Pilenil tab'
```



```
elif(h=='Heart attack'):

    medicines='Sorbitrate tab'

elif(h=='Varicose veins'):

    medicines='Polydocanal Injection , Sodium Tetradecyl Sulphate Injection
, Ibuprofin tab , Paracetomal tab'

elif(h=='Hypothyroidism'):

    medicines='Levothyroxine Soduim tab , Pyridoxin Hydrochloride
sustained released tab , Tirosint capsules , Levoxyl tab , Synthroid tab ,
Unithroid tab'

elif(h=='Hyperthyroidism'):

    medicines='Levothyroxine Soduim tab , Pyridoxin Hydrochloride
sustained released tab , Tirosint capsules , Levoxyl tab , Synthroid tab ,
Unithroid tab'

elif(h=='Hypoglycemia'):

    medicines='Glucose tab'

elif(h=='Osteoarthritis'):

    medicines='Glucosamine Sulphate tab , Ibuprofin tab , Naproxen tab'

elif(h=='Arthritis'):

    medicines='Glucosamine Sulphate tab , Ibuprofin tab , Naproxen tab'

elif(h=='(vertigo)Paroxysmal Positional Vertigo'):

    medicines='Sinarzine tab , Flunarzine tab'

elif(h=='Acne'):

    medicines='Clindamycin Phosphate & NicotinamideGel , Adapalene
Benzylperoxide Gel '

elif(h=='Urinary tract infection'):
```

```
medicines='Ofloxacin tab , Norfloxacin tab , Siprofloxacin tab ,  
Tinidazole tab'  
  
elif(h=='Psoriasis'):  
  
    medicines='Aloe vera creams , White soft paraffin Gel , Flucanazole tab  
(Optional if itchy) , Clobetasole Propionite and Salicylicacid Gel'  
  
elif(h=='Impetigo'):  
  
    medicines='Mupirosin Ointment , Retamulin Ointment , Amoxycillin  
Potassium Clavunate'  
  
    return  
  
(render_template('out.html',medicines=medicines,disease_details=h,fname=name,  
me,aadhaar=aadhaar,district=district,village=village,phone=phone,age=age,gender=gender))  
  
if __name__=="__main__":  
  
    app.run(debug=True)
```

5.6.2 Machine Learning Code

```
import pandas as pd  
  
from sklearn.tree import DecisionTreeClassifier  
  
from sklearn.metrics import accuracy_score  
  
import pickle  
  
data=pd.read_csv('Training.csv')  
  
X=data.iloc[:, :-1]  
  
Y=data.iloc[:, -1]  
  
X=X.values
```

```
Y=Y.values

classifier=DecisionTreeClassifier()

classifier.fit(X,Y)

test_data=pd.read_csv('Testing.csv')

X_test=test_data.iloc[:, :-1].values

Y_test=test_data.iloc[:, -1].values

Y_pred=classifier.predict(X_test)

print(accuracy_score(Y_pred,Y_test))

f=open('model.pkl','wb')

pickle.dump(classifier,f)

f.close()
```

CHAPTER 6

TESTING

6.1 Testing principles:

All tests should be traceable to customer requirements

- Tests should be planned long before testing begins.
- Testing should begin “in the small” and progress toward testing “in the large”
- Exhaustive testing is not possible.
- To be most effective, an independent third party should conduct testing.

6.2 Tests used in project:

6.2.1 Unit Testing:

The first level of testing is called as unit testing. Here different modulus are tested against the specifications produced during the design of the modulus. Unit testing is done to test the working of individual modulus. Unit testing focuses verification effort on the smallest unit of software design that is the module. Using procedural design description as a guide. Important control paths are tested to uncover errors within the boundaries of the module. The unit test is normally white box testing oriented and the step can be conducted in paralleled for multiple modules.

6.2.2 System Testing:

System testing is actually a series of different tests whose primary propose is to fully exercise to the computer based system. The various tests include recovery testing,

stress testing, and performed testing. Involves in-house testing of the entire. Its aim is to satisfy the user and system requirements of the client's specifications.

6.2.3 Acceptance Testing:

It is a pre-delivery testing in which entire system is tested at client's site on real world data and find errors.

6.3 Test Cases

Test Case ID	Test Case (symptoms)	Description	Expected Output	Actual Output	Result
TC1	itching nodal_skin_eruptions dischromic_patches skin_rash	Symptoms for fungal infection	Fungal Infection	Fungal Infection	Success
TC2	stomach_pain acidity ulcers_on_tongue vomiting cough chest_pain	Symptoms for GRED	GRED	GRED	Success
TC3	continuous_sneezing chills watering_from_eyes	Symptoms for Allergy	Allergy	Allergy	Success
TC4	weakness_in_limbs neck_pain dizziness loss_of_balance	Symptoms for Cervical spondylosis	Cervical spondylosis	Cervical spondylosis	Success

Table 6.3.1 Table of Positive test cases

Test Case ID	Test Case (symptoms)	Description	Expected Output	Actual Output	Result
TC1	acidity indigestion headache blurred_and_distorted_vision	Symptoms for Migraine	Migraine	GERD	failure
TC2	muscle_weakness stiff_neck swelling_joints movement_stiffness	Symptoms for Arthritis	Arthritis	Malaria	failure
TC3	fatigue high_fever nausea constipation	Symptoms for Typhoid	Typhoid	Chicken pox	failure

Table 6.3.2 Table of negative test cases

CHAPTER 7

RESULTS

7.1 Screenshots

Running python script in terminal

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\HP\Desktop\AWS_ML> python app.py
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 102-226-211
```

Figure 7.1: Running code on Terminal

Web Page

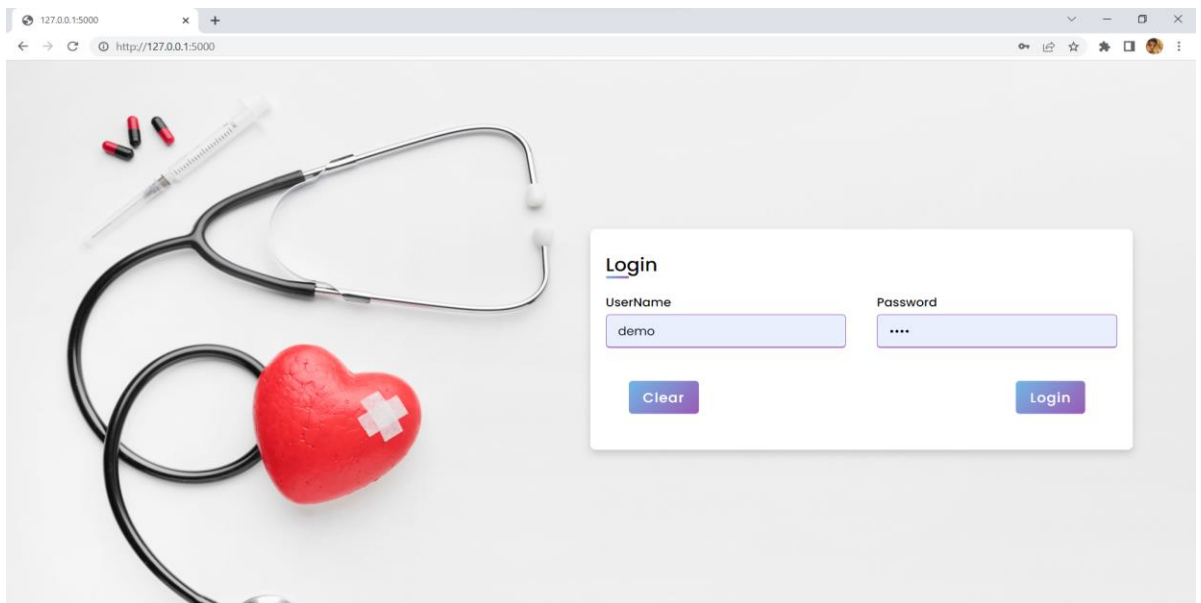


Figure 7.2: Login page

Giving symptoms of the patient-1

Fill The Details

Full Name: Anupama Parvathareddy

Aadhaar Number: 603682309113

District: Guntur

Village Name: k.b.palem

Phone Number: 8897213281

Age: 21

Gender: ☐ Male ☒ Female

Symptoms

Symptoms: Type Atleast 2 letters

Note:

1. Search for Symptom from box, select the symptom and click "Confirm" button.
2. If you want to delete the symptom from Textarea select the same symptom from box and click the "Delete" button.

itching
nodal_skin_eruptions
dischromic_patches
skin_rash

Confirm Delete Evaluate

Figure 7.3: Giving symptoms of patient-1

Disease recognized by ML through API gateway for patient-1

Patient Report

Full Name: Anupama Parvathareddy

Aadhaar Number: 603682309113

District: Guntur

Village Name: k.b.palem

Phone Number: 8897213281

Age: 21

Gender: female

Disease Details

Fungal infection

Prescribed Medicines

Fluconazole tab , if chronic - voriconazole tab (Fungal Infection)

Save

Figure 7.4: Disease recognized for patient-1

Giving symptoms of the patient-2

The screenshot shows a web browser window with two main sections: "Fill The Details" and "Symptoms".

Fill The Details:

- Full Name:** Jaya Prakash Naryanaa
- Aadhaar Number:** 568723416573
- District:** Prakasam
- Village Name:** chirala
- Phone Number:** 06302442682
- Age:** 22
- Gender:** Male (selected), Female
- Clear** button

Symptoms:

- Symptoms:** Type Atleast 2 letters
- Note:**
 1. Search for Symptom from box, select the symptom and click "Confirm" button.
 2. If you want to delete the symptom from Textarea select the same symptom from box and click the "Delete" button.
- Symptom List:**
 - stomach_pain
 - ulcers_on_tongue
 - acidity
 - vomiting
 - cough
 - chest_pain
- Buttons:** Confirm, Delete, Evaluate

Figure 7.5: Giving symptoms of patient-2

Disease recognized by ML through API gateway for patient-2

The screenshot shows a web browser window displaying a "Patient Report".

Patient Report:

- Full Name:** Jaya Prakash Naryanaa
- Adhaar Number:** 568723416573
- District:** Prakasam
- Village Name:** chirala
- Phone Number:** 06302442682
- Age:** 22
- Gender:** male
- Disease Details:** GERD
- Prescribed Medicines:** Raveprazole tab , Lansoprazole tab , Esomeprazole tab , Dexlansoprazole tab
- Save** button

Figure 7.6: Disease recognized for patient-2

CHAPTER – 8

CONCLUSION

In this project we have restricted the foreign developers to access the code directly for their requirements rather we have provided an interface, by which they can get their job done through API. we have used micro-services for healthcare-related services and we have developed an ML model by which we can predict the diseases accurately. We have trained the model with several datasets for more accuracy.

CHAPTER - 9

FUTURE SCOPE

In future, we are planning to apply lightweight protocols like MQTT (Message Queuing Telemetry Transport), WebSockets for communication, and in this way, we improve our accuracy and reduce the cost of the project maintenance. We plan to extend the system and the dataset to cover more data types.

CHAPTER - 10

REFERENCES

- A. Mir, S.N. Dhage, in 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) (IEEE, 2018), pp. 1–6
- Nicholas Brown, AWS: Amazon Elastic Compute Cloud (EC2): Guide for Beginners (2017)
- Jurg van Vliet, Flavia Paganelli, Programming Amazon EC2: Survive your Success (2018)
- Y. Khourdifi, M. Bahaj, Heart disease prediction and classification using machine learning algorithms optimized by particle swarm optimization and ant colony optimization, Int. J. Intell. Eng. Syst. 12(1), 242 (2019)
- S. Vijayarani, S. Dhayanand, Liver disease prediction using svm and naïve bayes algorithms, International Journal of Science, Engineering and Technology Research (IJSETR) 4(4), 816 (2015).
- S. Mohan, C. Thirumalai, G. Srivastava, Effective heart disease prediction using hybrid machine learning techniques, IEEE Access 7, 81542 (2019)
- T.V. Sriram, M.V. Rao, G.S. Narayana, D. Kaladhar, T.P.R. Vital, Intelligent parkinson disease prediction using machine learning algorithms, International Journal of Engineering and Innovative Technology (IJEIT) 3(3), 1568 (2013)

- A.S. Monto, S. Gravenstein, M. Elliott, M. Colopy, J. Schweinle, Clinical signs and symptoms predicting influenza infection, Archives of internal medicine 160(21), 3243 (2000)
- R.D.H.D.P. Sreevalli, K.P.M. Asia, Prediction of diseases using random forest classification algorithm.
- D.R. Langbehn, R.R. Brinkman, D. Falush, J.S. Paulsen, M. Hayden, an International Huntington's Disease Collaborative Group, A new model for prediction of the age of onset and penetrance for huntington's disease based on cag length, Clinical genetics 65(4), 267 (2004)
- K. Kourou, T.P. Exarchos, K.P. Exarchos, M.V.Karamouzis, D.I. Fotiadis, Machine learning applications in cancer prognosis and prediction, Computational and structural biotechnology journal 13, 8 (2015)
- T. Karayilan, O. Kılıç, in " 2017 International Conference on Computer Science and Engineering (UBMK) (IEEE, 2017), pp. 719–723