**Chapter 5**

**Implementation and testing**

**Coding**

**Server.py: This file is user to serve api and handling server side data/backend.**

from flask import Flask, render\_template, request,send\_file

from send\_mail import \*

from random import randint

from DB\_module import \*

from socket import gethostname,gethostbyname

import cert

import os

app=Flask(\_\_name\_\_)

session\_otp\_id=[]

@app.route("/")

def start():

return render\_template("index.html")

@app.route("/user")

def user():

return render\_template("/user\_side/user\_index.html")

@app.route("/user\_login")

def userl():

return render\_template("/user\_side/user\_login.html")

@app.route("/user\_panel")

def user\_panel():

return render\_template("/user\_side/user\_panel.html")

@app.route("/add\_member\_panel")

def add\_member\_panel():

return render\_template("/user\_side/add\_member.html")

@app.route("/send\_otp/",methods=["POST"])

def controll\_car():

global session\_otp\_id

req=request.get\_json()

session\_id=randint(1111111,99999999999999)

otp=send\_email\_otp(req["email"])

if (otp=="False"):

return "False"

session\_otp\_id.append({"email":req["email"],"session\_id":session\_id,"otp":otp})

return {"otp":str(otp),"session\_id":session\_id}

@app.route("/login\_admin")

def login\_admin():

return render\_template("login\_admin.html")

@app.route('/vadapav/<filename>')

def download\_image(filename):

# Set the filename and mimetype of the file

filename="static/certificates/"+filename+"\_certificate.pdf"

print(filename)

if(os.path.isfile(filename)==True):

mimetype = 'application/pdf'

return send\_file(filename, mimetype=mimetype, as\_attachment=True, attachment\_filename='downloaded-pdf.pdf')

else:

return "File not exists"

#filename="static/certificates/"+filename

# Use the send\_file function to send the file

@app.route("/admin\_dashboard")

def admin\_dashboard():

return render\_template("new\_admin\_page.html")

@app.route("/doctor\_dashboard")

def doctor\_dashboard():

return render\_template("temp.html")

@app.route("/add\_doctor\_form")

def add\_doctor\_form():

return render\_template("add\_doctor\_form.html")

@app.route("/add\_patient\_form")

def add\_patient\_form():

return render\_template("add\_patient\_form.html")

@app.route("/treat\_parent")

def treat\_parent():

return render\_template("patients\_tratment\_form.html")

@app.route("/requests")

def requestststtstttst():

return render\_template("request\_member.html")

@app.route("/add\_vaccine\_form")

def add\_vaccine\_form():

return render\_template("add\_vaccine\_form.html")

@app.route("/update\_vaccine")

def update\_vaccine():

return render\_template("update\_vaccine.html")

@app.route("/certificate")

def certificate():

print("certificate")

return render\_template("certificate.html")

# @app.route("/center\_registration\_form")

# def crf():

# return render\_template("center\_registration\_form.html")

@app.route("/Main\_content")

def mc():

return render\_template("Main\_content.html")

# @app.route("/center\_panel")

app.run(host=gethostbyname(gethostname()))

**send\_mail.py : This file is used to send otp and other mails**

from random import randint

from email.message import EmailMessage

import smtplib

def send\_email\_otp(to):

try:

otp=randint(1000,99999)

body="OTP From e\_vaccination portal\nYour OTP is: "+str(otp)

subject="e-Vaccination"

msg = EmailMessage()

msg.set\_content(body)

msg['subject']=subject

msg['to'] = to

user = "sarveshgandhere2002@gmail.com"

msg['from']=user

password = "kburswetbalyszew"

server = smtplib.SMTP("smtp.gmail.com",587)

server.starttls()

server.login(user,password)

server.send\_message(msg)

server.quit()

except:

return "False"

else:

return str(otp)

**DB\_module.py : This file is used to do database related operations.**

import mysql.connector

from datetime import date

import random

class handle\_data():

\_\_mydb=mysql.connector.connect(host="localhost",user="root",password="1234",database="vaccination")

\_\_mycursor = \_\_mydb.cursor()

def save\_new\_user(self,mailid,name):

#query="insert into login\_data values (%s, %s, %s, %s, %s)"

query="select \* from login\_data where mailid=%s"

value=(mailid,)

self.\_\_mycursor.execute(query,value)

result=self.\_\_mycursor.fetchall()

if result==[]:

query="insert into login\_data values(%s,%s)"

value=(mailid,name,)

self.\_\_mycursor.execute(query,value)

self.\_\_mydb.commit()

print("data inserted")

return "new User"

else:

query="select \* from centers where mailid=%s"

value=(mailid,)

self.\_\_mycursor.execute(query,value)

result=self.\_\_mycursor.fetchall()

return result

**cert.py: This file is used to create certificates.**

import os

import base64

import io

from PIL import Image, ImageDraw, ImageFont

import send\_mail

from socket import gethostname,gethostbyname

def generate\_certificate(name,diseas,doctor,to):

# Load the certificate template image

font\_size=75

cert\_template\_path = "D:\\Projects\\OPD\\static\\certificate.png"

output\_folder = "D:\\Projects\\OPD"

font= "D:\\Projects\\External\_downloads\\Fonts\\dino\_and\_friend\\Aboreto-Regular.ttf"

line\_font="D:\\Projects\\External\_downloads\\Fonts\\dino\_and\_friend\\Raleway-Black.ttf"

line=diseas+" has been declared fit."

line\_font\_size=45

doctor\_font="D:\\Projects\\External\_downloads\\Fonts\\dino\_and\_friend\\Raleway-VariableFont\_wght.ttf"

doctor\_font\_size=45

cert\_template = Image.open(cert\_template\_path)

# Define the text to be added to the certificate

text = name

# Determine the font size and thickness based on user input

font\_scale = int(font\_size \* 0.75) # Pillow font size is in points, not pixels

font\_thickness = int(font\_size \* 0.05) # You can adjust this as needed

line\_font\_scale=int(line\_font\_size\*0.75)

doctor\_font\_scale=int(doctor\_font\_size\*0.75)

#line\_font\_thickness=

# Load the font

if font.lower() == "default":

font = ImageFont.load\_default()

else:

# Load a custom font file

font\_path = os.path.join(os.path.dirname(os.path.abspath(\_\_file\_\_)), font)

font = ImageFont.truetype(font\_path, size=font\_scale)

line\_font\_path=os.path.join(os.path.dirname(os.path.abspath(\_\_file\_\_)), line\_font)

line\_font = ImageFont.truetype(line\_font\_path, size=line\_font\_scale)

doctor\_font\_path=os.path.join(os.path.dirname(os.path.abspath(\_\_file\_\_)), doctor\_font)

doctor\_font = ImageFont.truetype(doctor\_font\_path, size=doctor\_font\_scale)

# Get the size of the text

text\_size = font.getsize(text)

line\_size= line\_font.getsize(line)

doctor\_size=doctor\_font.getsize(doctor)

# Calculate the coordinates of the text

x = (cert\_template.width - text\_size[0]) // 2

y = (cert\_template.height - text\_size[1]) // 2

y=320

x1=(cert\_template.width - line\_size[0]) // 2

y1=435

x2=275-(doctor\_size[0]//2)

y2=525

# Add the text to the image

print(line)

draw = ImageDraw.Draw(cert\_template)

draw.text((x, y), text, font=font, fill="black")

draw.text((x1,y1),line,font=line\_font,fill=(0,0,0))

draw.text((x2,y2),doctor,font=doctor\_font,fill=(0,0,0))

# Save the certificate with a new filename

#output\_path = os.path.join(output\_folder, f"{name}\_certificate.png")

#cert\_template.save(f"{name}\_certificate.png")

cert\_template.save("static/certificates/"+name+"\_certificate.pdf")

#send\_mail.send\_certificate\_link("http://"+gethostbyname(gethostname())+":5000/certificate/"+name+"\_certificate.pdf",to)

send\_mail.send\_certificate\_link(gethostbyname(gethostname())+":5000/certificate",to)

#print(f"Certificate for '{name}' saved to {}")

#print(get\_font("D:\\Projects\\External\_downloads\\Fonts\\static"))

name = "John Doe"

font\_size = 75

#font = "D:\\Projects\\External\_downloads\\Fonts\\static\\Raleway-Black.ttf"

#pre(name, font\_size, "Dino And Friend-Texture1.ttf", cert\_template\_path)

#generate\_certificate(name, "headech","Sarvesh Gandhere")

**Testing**

In the testing part, the software will be tested on a different basis. Three testing approaches will be used to check whether the software is working efficiently or not. The three testing methodologies used are

1. Unit Testing: In this type of testing, each unit i.e. each function of the system

will be tested. This testing will be done at the time of development. If any

error occurs then it will be immediately resolved. In our case, some of the

units of our system are the registration form, login form, profile display, etc.

2. Integration Testing: Combining multiple units of a system makes a module.

A module is a collection of code for related units. Here we will check whether

the software works efficiently after combining the unit. The modules in this

system will be the candidate module, company module, and system module.

3. System Testing: Here, we will check whether the entire system runs properly

after combining all the modules without any bugs. This testing will be done

after the development of the code.

4. Compatibility Testing: In this testing, we test whether the web application

works efficiently on different browsers and different operating systems.

Testing is done by running the website on different browsers including

Firefox, Microsoft Edge, and Chrome.

5. Performance Testing: In this testing, we check that how well a web

application can run at different network speeds and what is the behavior of the

website under normal load and peak load.

6. Database Testing: Here we test whether the necessary queries are performed accurately in less time or not. Whether the test data retrieved from the database is accurately displayed on the system or not.

7. Usability Testing: In this testing, we check whether the UI/UX design of the

web application functions accurately and whether the user is able to easily

work on the system.

**Chapter 6**

**6.1 Test Reports**

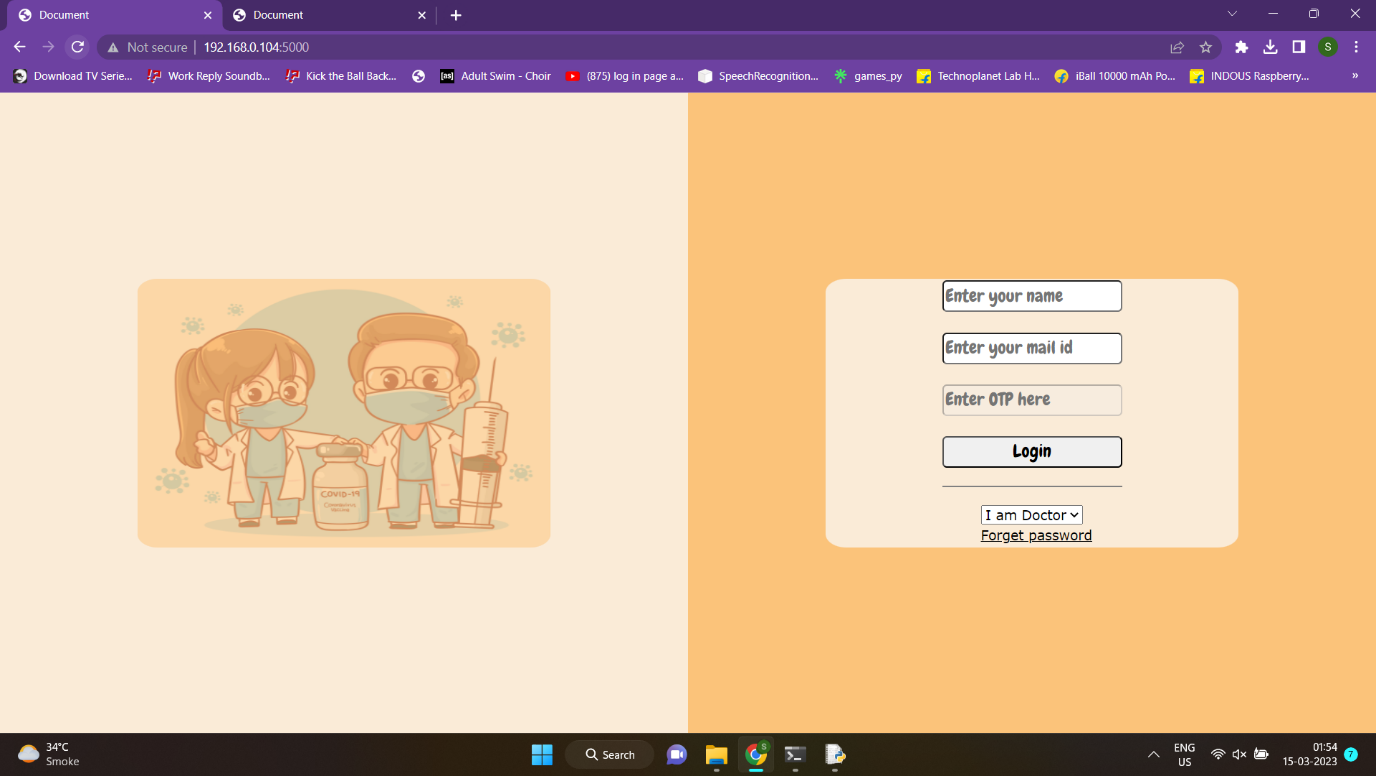
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test case id | Test case name | Description | Input | Excepted output | Actual output | Status |
| T1 | Login | Checking whether login information is correct or not | mail id  OTP | Doctors dashboard/Admin dashboard should display | Doctors dashboard/Admin dashboard Appears | Pass |
| T2 | Admin page | Checking whether all details in admin panel is displayed or not | NIL | Admin Panel display should have correct data | Admin Panel display have correct data | Pass |
| T3 | Add doctor | Checking whether all details are correct or not | Doctors details | Doctor should be added | Doctor gets added | Pass |
| T4 | Add patient | Checking whether all details are correct or not | Patient details | Patient should get added | Patient get added | Pass |
| T5 | Add vaccine | Checking whether all details are correct or not | Vaccine details | Vaccine should get added | Vaccine get added | Pass |
| T6 | Show vaccine | Displaying vaccines | NIL | Vaccine details should displayed | Vaccine details get displayed | Pass |
| T7 | Requests | Checking for any request to add patient | NIL | Requests page should display | Request page get display | Pass |
| T8 | Show doctors | Displaying doctors details | NIL | Doctors should get displayed | Doctors get displayed | Pass |
| T9 | Show patient | Displaying patient details | NIL | Patient details should get displayed | Patient details get displayed | Pass |
| T10 | Update vaccine | Updating quantity of vaccine | Quantity to be added | Quantity of vaccine should get update | Quantity of vaccine get update | Pass |
| T11 | Login as doctor | Login in system using doctors credential | Doctors credential | Doctors page should get display | Doctors page get displayed | Pass |
| T12 | Patient details to doctor | Showing patient details to doctor | NIL | Patient details should be displayed | Patient details get displayed | Pass |
| T13 | Add patient | Adding patient with some input | Patient details | Adding patient from doctors side | Adding patient from doctors side | Pass |
| T14 | Start treatment | Starting patient treatment | NIL | Patients medical details and treatment form should appear | Patients medical details and treatment form get appear | Pass |
| T15 | Treatment update | Updating patient medical details | Patient medical details | Patient details should get update | Patient details get updated | Pass |
| T16 | User page | Checking whether user page is visible or not | NIL | User page should displayed | User page get displayed | Pass |
| T17 | User login | Checking for user credentials | User data | Otp page should be appear | Otp page get appear | Pass |
| T16 | User Login | Checking whether users otp is correct or not | OTP | User page should get displayed | User page get displayed | Pass |
| T17 | User login | After successfully login process | NIL | User page should get display with correct information | User page get displayed with correct information | Pass |
| T18 | Download certificate | Checking whether certificate gets download or not | NIL | Certificate should get download | Certificate gets download | Pass |
| T19 | Add member page | Checking whether add member page is working | NIL | Add member page should get displayed | Add member page get displayed | Pass |
| T20 | Add member | Adding new family member | NIL | Family members request should send to admin | Family members request send to admin | Pass |
| T21 | Status of new member request | Checking whether status of new member request is going through mail | NIL | User should get mail about status | User get mail about status | Pass |
| T22 | Certificate mail | Checking whether user is getting certificate mail | NIL | User should get mail about certificate generation | User get mail about certificate generation | Pass |

**6.2 User Documentation**

In this section, screenshots of all the parts of the web application along with information related to it are given. This section is divided into 3 parts Admin module, Doctor module, User module.

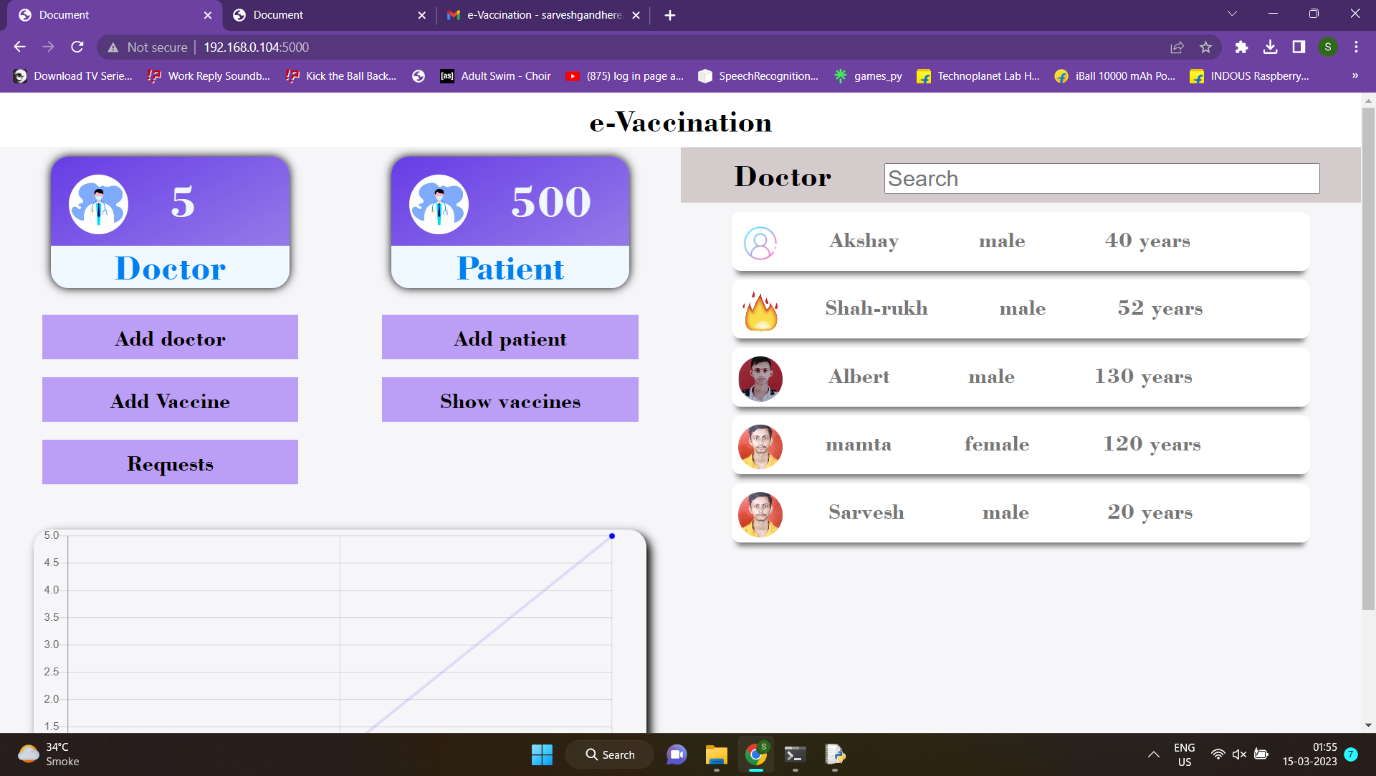
**Admin module**

*Login Page doctor/Admin*

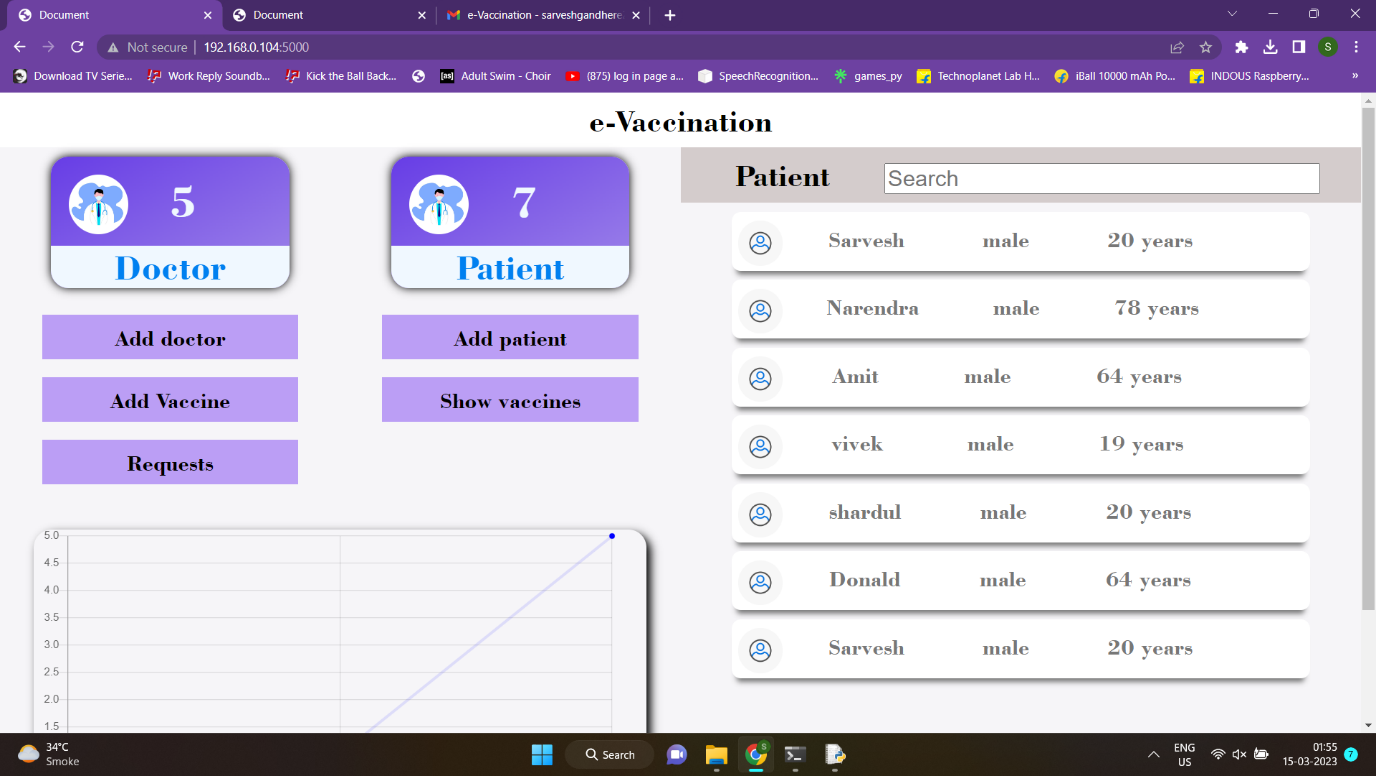
****

This page is use to login for doctor and admin.

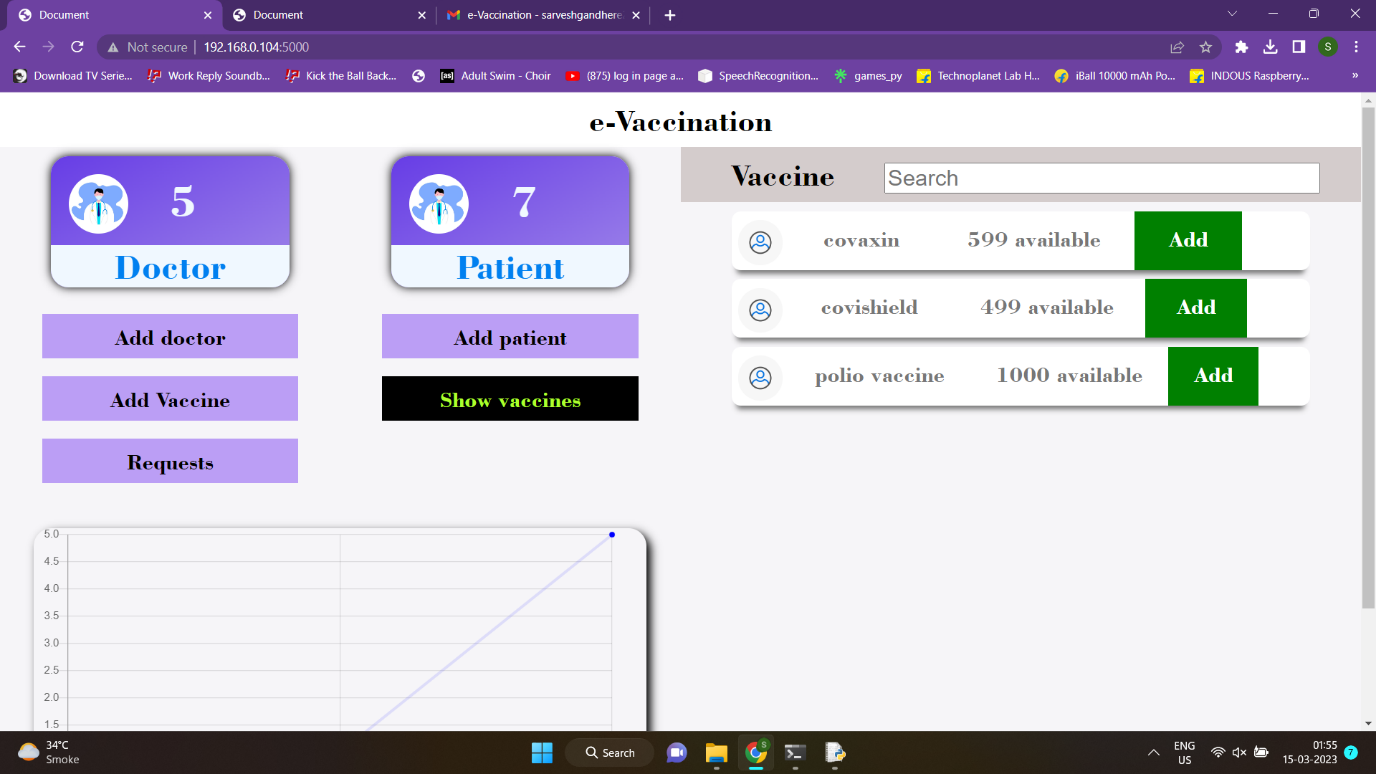
*Admin Page*

**

When Admin login this page will appear. This page is divide into 3 components. Control component where all buttons are located. Graph component where graph represents patient/day. Vie component where patient data, doctor data and vaccine data displays.

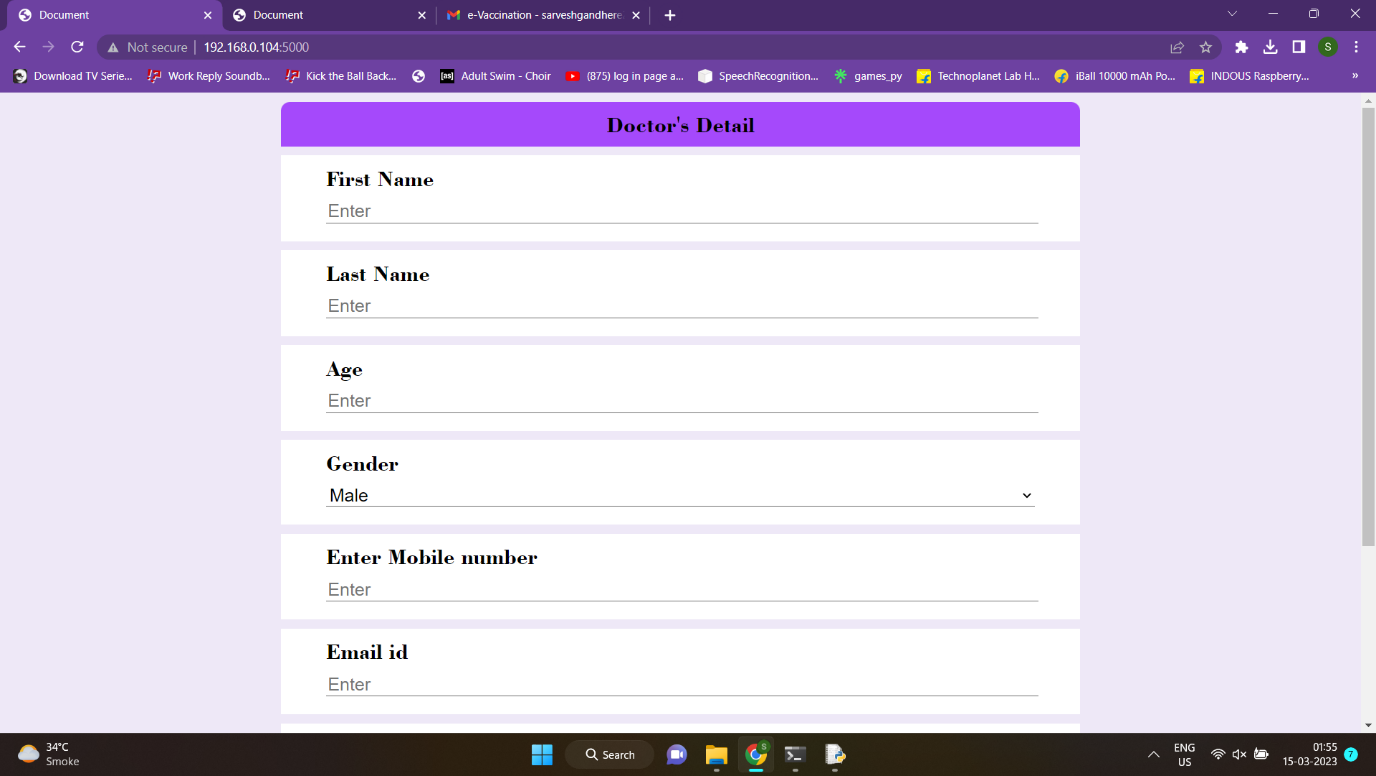


If we click on patient button all patient data is display.



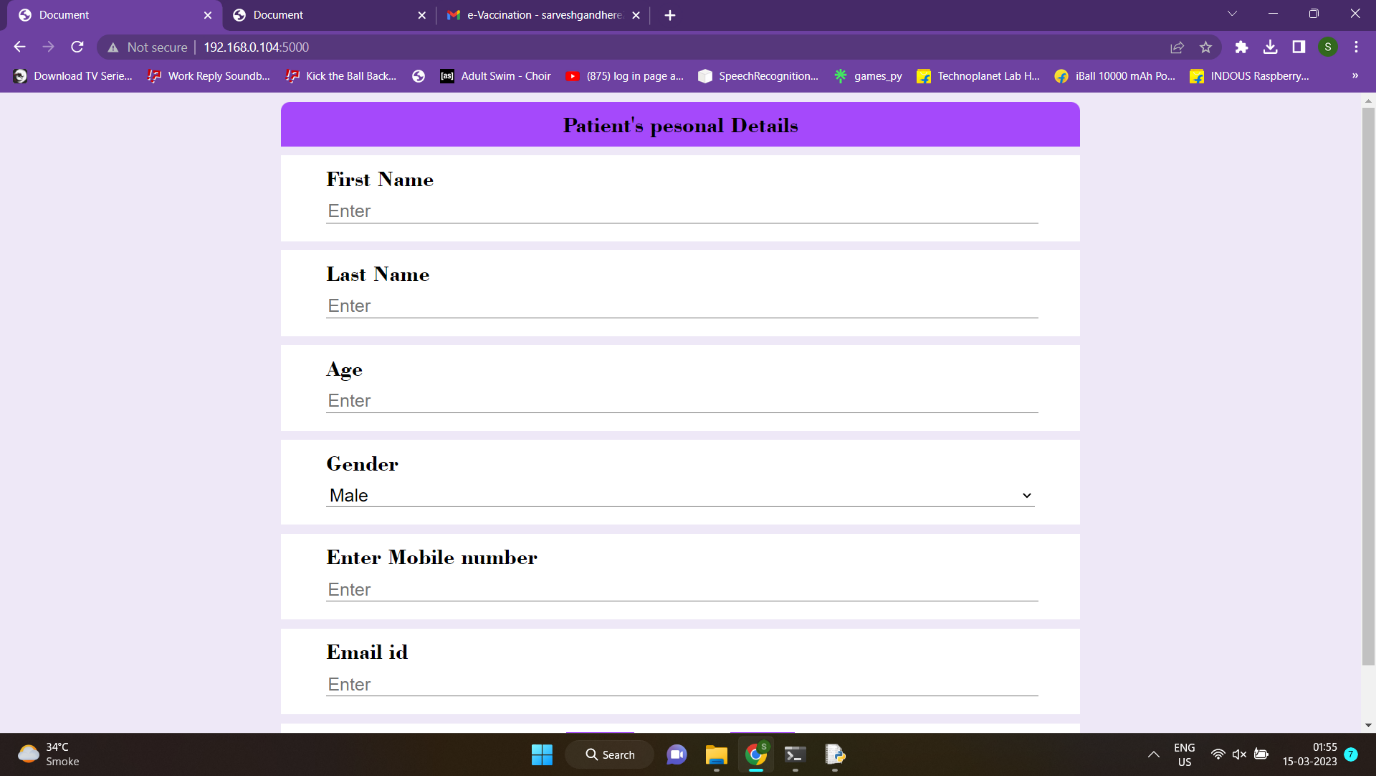
If we click on show vaccine button all vaccine data is display.

*Doctors form*



When admin clicks on add doctors then this page will appear.

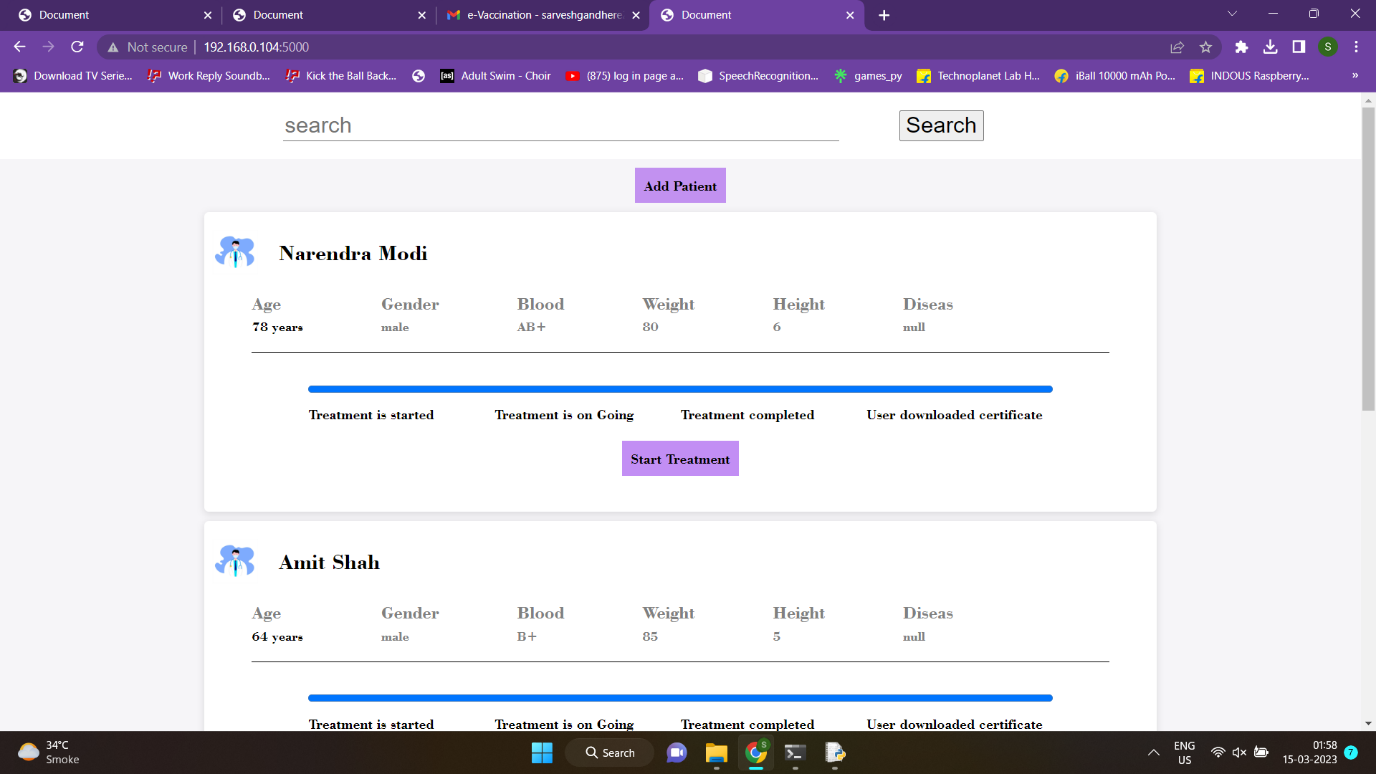
*Patient form*

**

When Admin clicks on add patient this form will appear.

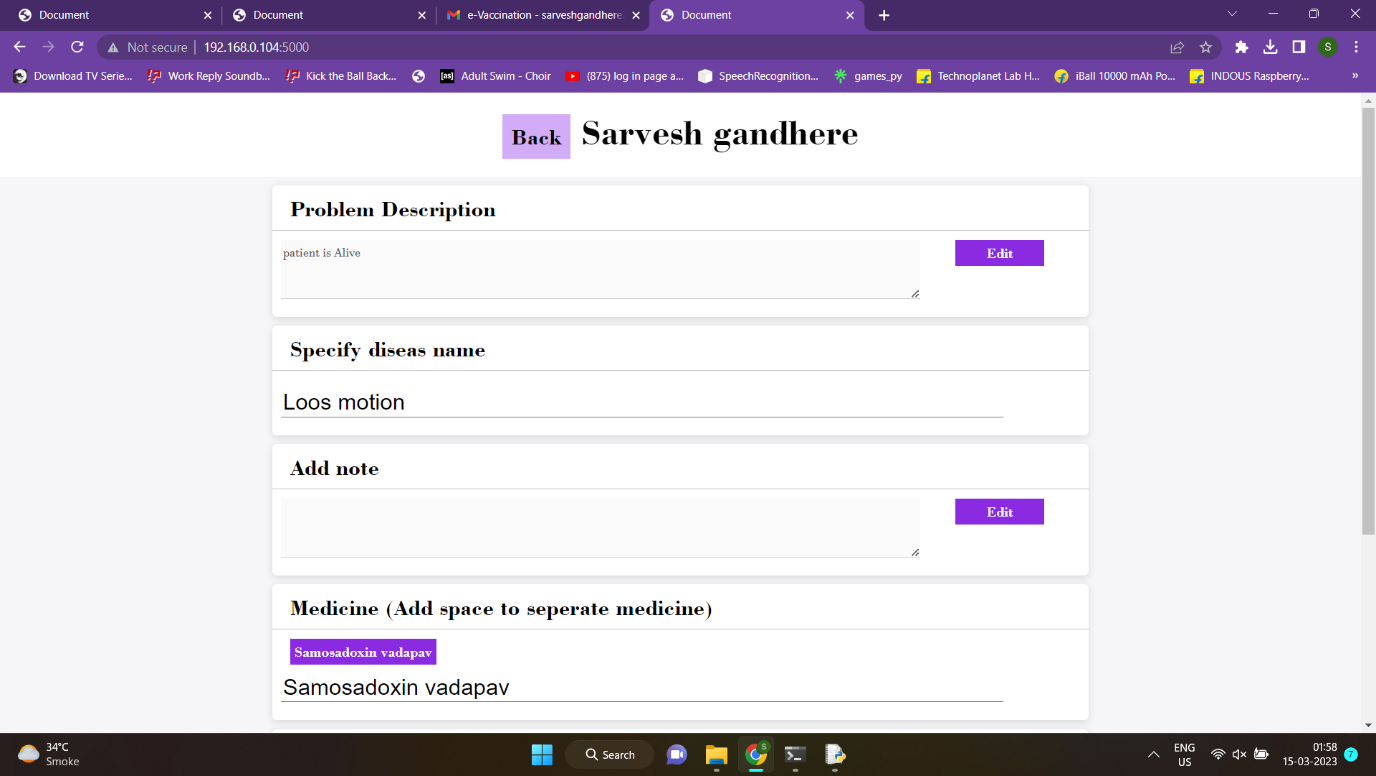
**Doctor module**

*Doctors page*



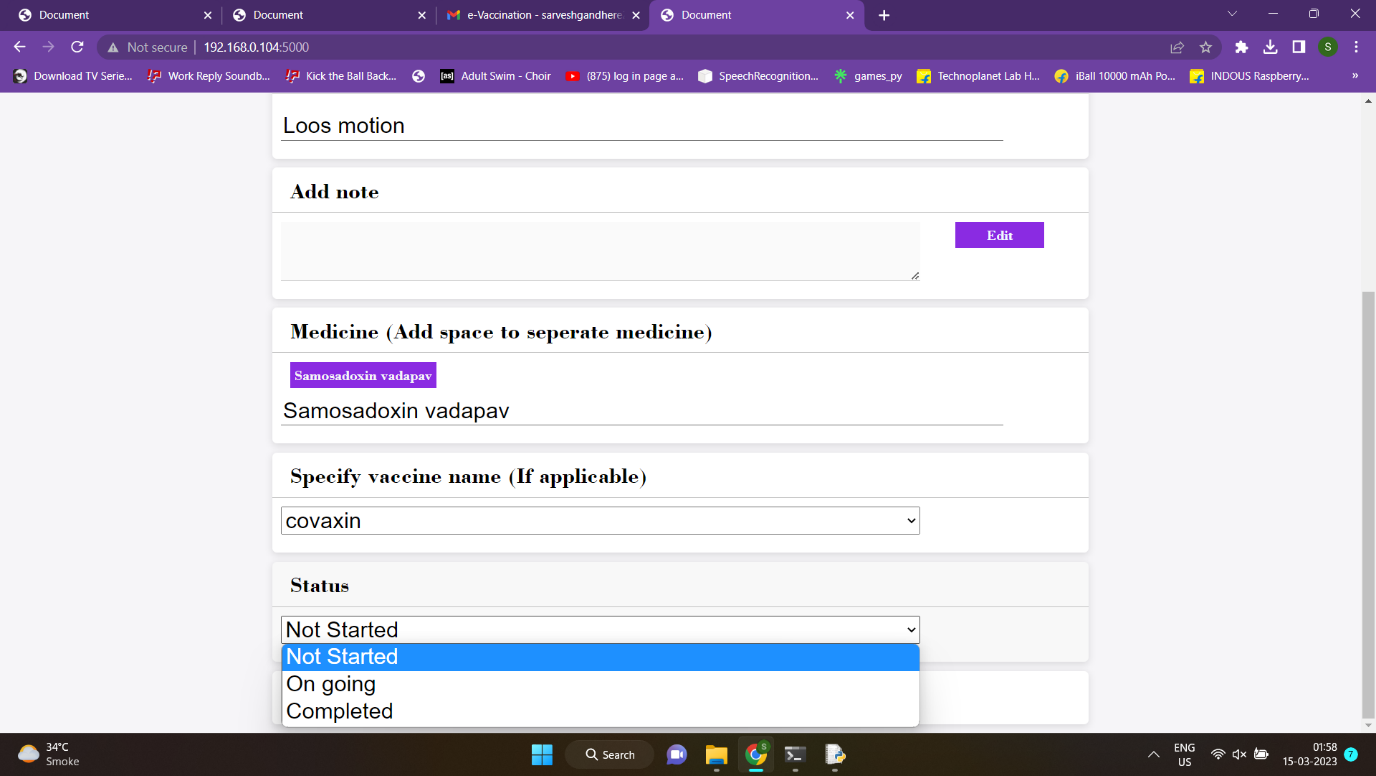
When doctor login this page will appear.

*Treatment form*

**

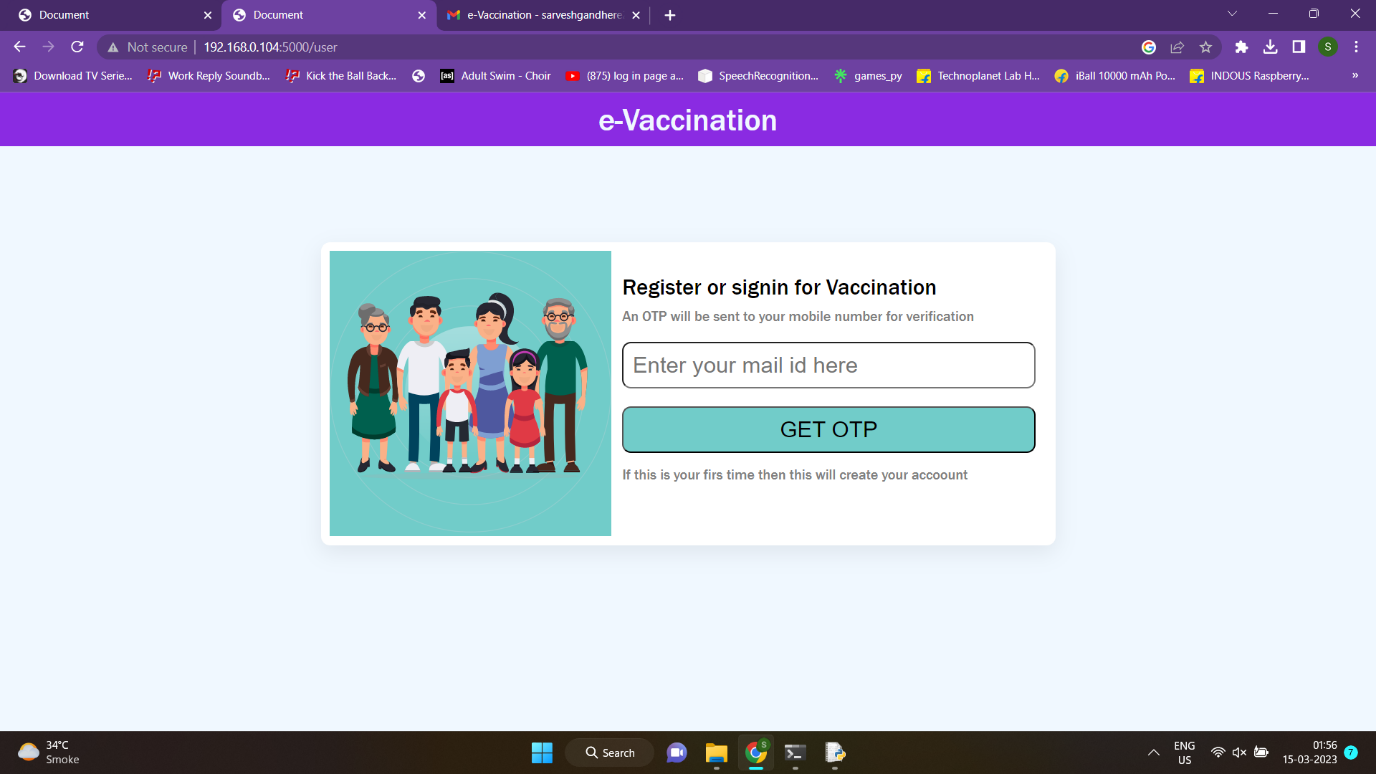
When doctors start treatment of any patient then this form will appear after clicking on start treatment button.

*treatment form*



**User module**

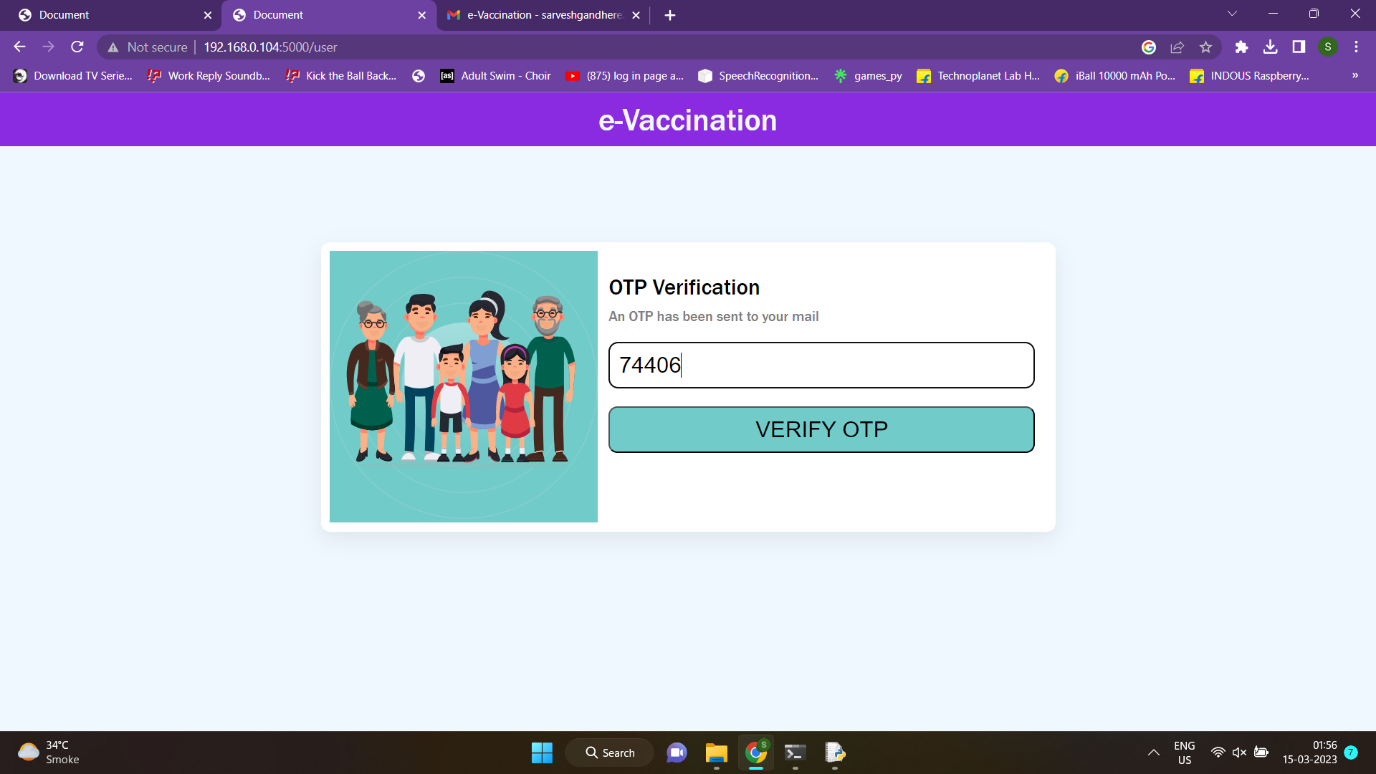
*User login-mail form*

**

This form used to login as a user. Where after entering email user will get otp.

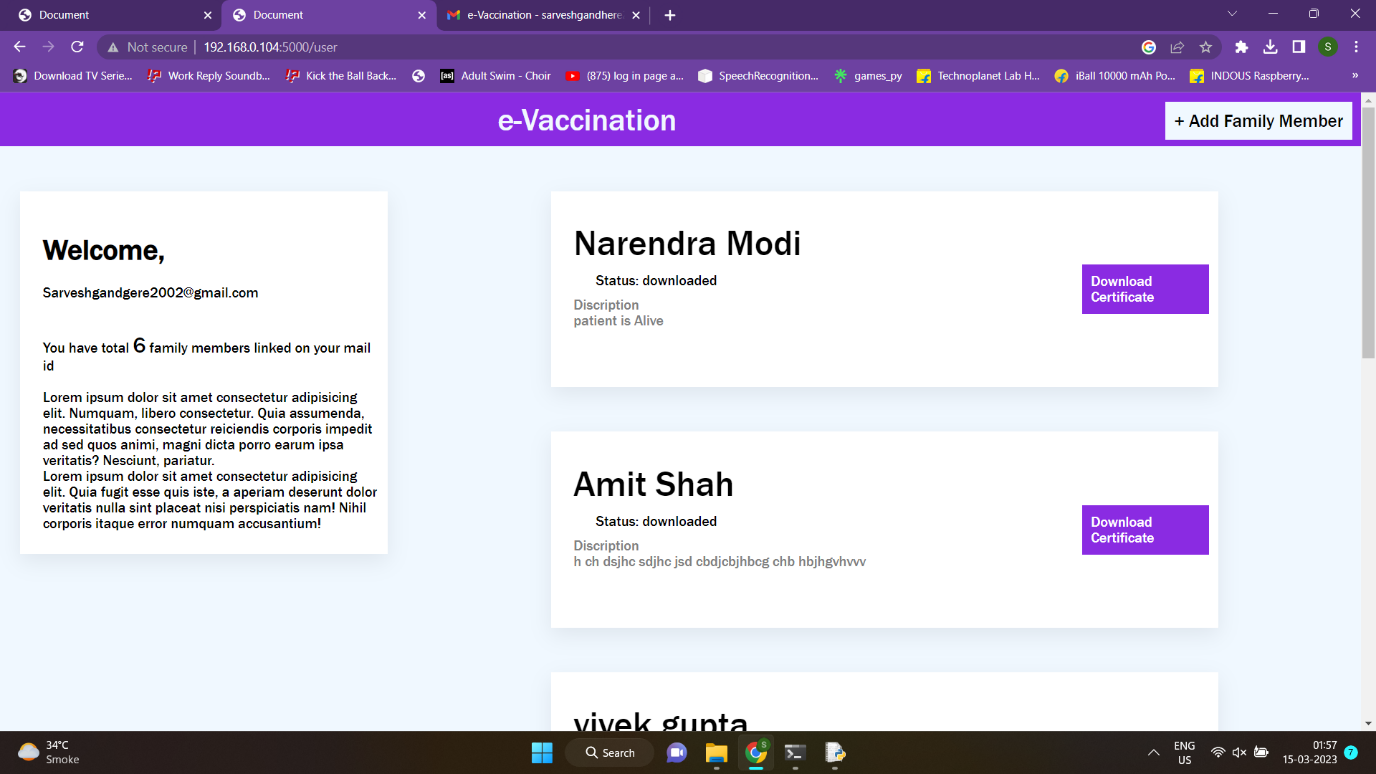
If user is new then his account will be created automatically.

*User login-otp form*

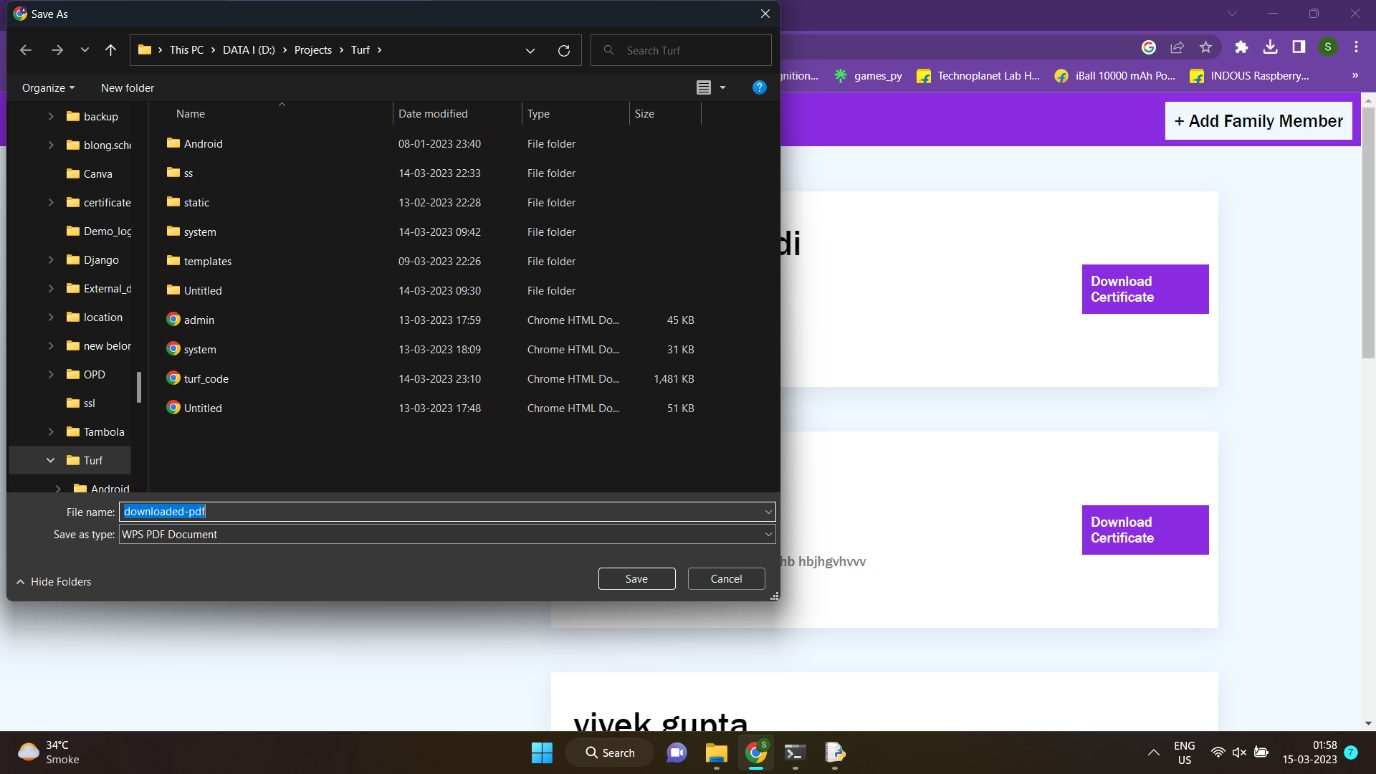
**

Here user can insert otp.

*User page*

**

In this page user can add his family member and download his certificates and keep a track of his treatment.

User page

If user clicks on download certificate button.

**Chapter 5**

**Conclusions**

**Conclusion:**

Our vaccination system provides a user-friendly and efficient platform for patients to apply for treatment and obtain their medical certificates.

The system features a simple login process and a comprehensive database of doctors and their specializations.

Our system successfully meets the objectives of the project by providing a paperless solution for managing vaccination records.

The system has several advantages over traditional paper-based systems, such as improved data accuracy, increased accessibility, and faster record retrieval.

**Limitations:**

During the development process, we encountered technical issues resulting in system crashes and slow response times.

Some features and functionalities could not be implemented due to technical or time constraints.

There may be potential issues with system security that could compromise patient data.

The system is limited by the availability of internet connectivity and may not be accessible to all users in remote areas.

**Future Scope:**

Improvements can be made to the system's user interface to make it more visually appealing and user-friendly.

Additional features, such as telemedicine and online appointment scheduling, can be added to enhance the system's usefulness.

The system can be scaled up to cater to a larger user base by partnering with hospitals and clinics.

Collaborations can be established with healthcare providers to further develop and improve the system's functionalities.