DISTRIBUTED HEALTHCARE SYSTEM

# Group 4

# Priyanka Subramanyam

# Karthika MadhavanPillai Sasidharan Nair

# Srividhya Prakash Rangarajan

Contents

[**1.Introduction 3**](#_Toc513755041)

[**Use cases/Personas: 3**](#_Toc513755042)

[**2.Design/Architecture 4**](#_Toc513755043)

[**Description: 4**](#_Toc513755044)

[**I. Nginx Reverse Proxy Server 4**](#_Toc513755045)

[**II. WINSOCK Implementation 8**](#_Toc513755046)

[**III. Client in BSD Sockets 10**](#_Toc513755048)

[**3.Results and analysis 12**](#_Toc513755049)

[**4.Future work 21**](#_Toc513755076)

[**5.Conclusion 21**](#_Toc513755081)

[**6.References 21**](#_Toc513755082)

[**7.Contribution 21**](#_Toc513755083)

**8.Appendix……………………………………………………………………………………………………………………………………………………23**

# 1.Introduction

The fact that the Medical industry is in dire need of proper and secure technical services is a matter of concern as it constitutes an extremely important area in any country. The main goal of this project is to help reduce the gap and design a Distributed Healthcare system that would be a one stop destination for all kinds of users say patients, doctors, insurance agents and the system admin.

The objective of this paper is to summarize the different kinds of functionalities available to each user in this system: ‘Eir Hospitals’.

## Use cases/Personas:

1.    Brandon: Brandon is 32 years old and lives in California. He would like to get his regular health check-up done. Using the EIR Hospitals system he can book appointment with a doctor of his choice at any available date and time. Once his check up and scan is done his medical reports will be mailed to him. He can also cancel any appointment that he made and resend his reports to himself incase he needs it in the future.

A person standing in a room

Description generated with very high confidence

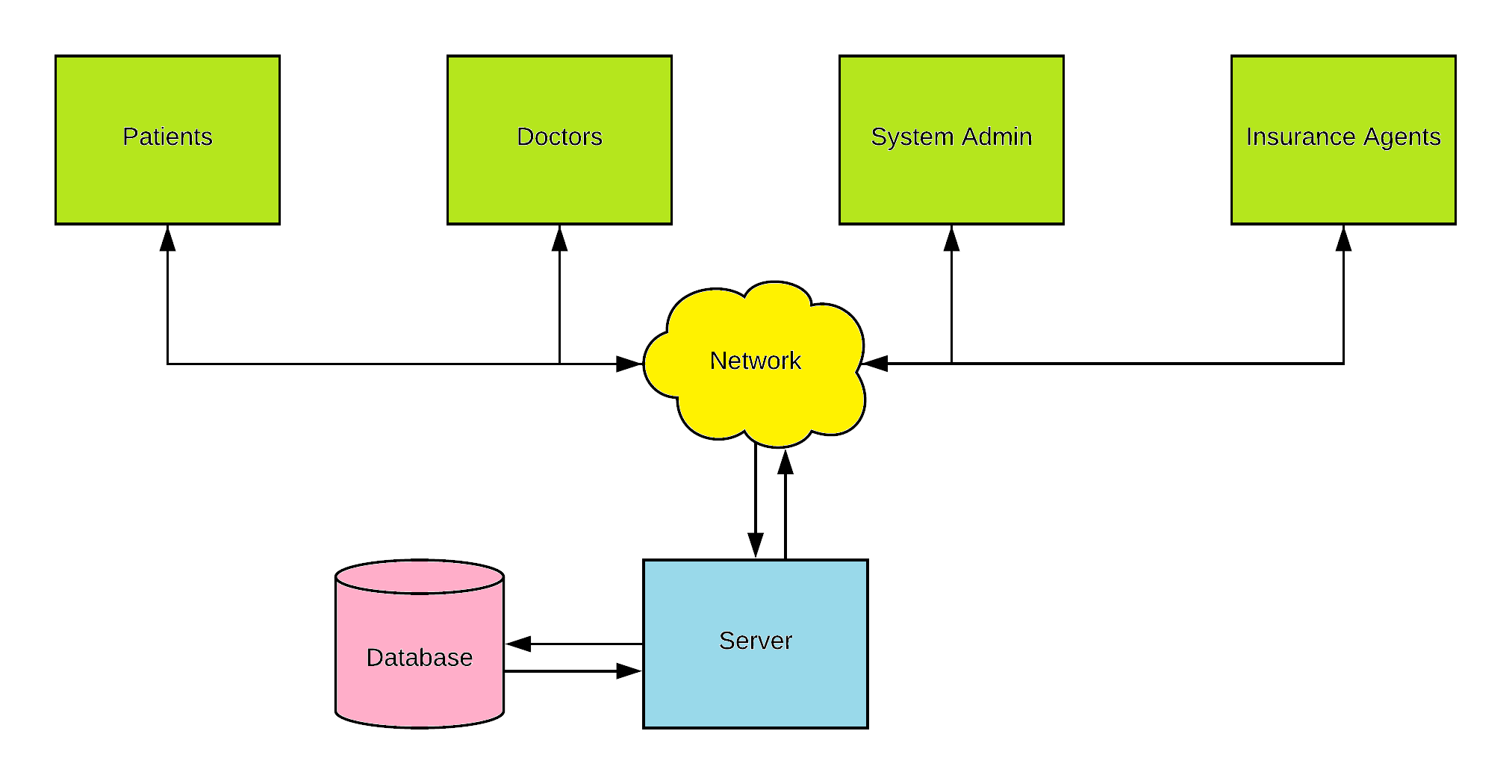
2.   Eli Sanchez: Dr.Sanchez working at EIR Hospitals would like to see all of her appointments at a particular date. She would also like to check a patient’s medical history and upload any scanned reports. Through the system she can not only do this but also email it to them personally.

Two people looking at the camera

Description generated with high confidence

# 2.Design/Architecture

The following diagram represents the architecture of the system.



The main components used are outlined below.

A screenshot of a cell phone

Description generated with very high confidence

# **Description:**

# Nginx Reverse Proxy Server

In [computer networks](https://en.wikipedia.org/wiki/Computer_network), a reverse proxy is a type of [proxy server](https://en.wikipedia.org/wiki/Proxy_server) that retrieves resources on behalf of a [client](https://en.wikipedia.org/wiki/Client_(computing)) from one or more [servers](https://en.wikipedia.org/wiki/Server_(computing)). These resources are then returned to the client as if they originated from the Web server itself. Unlike a [forward proxy](https://en.wikipedia.org/wiki/Proxy_server#Forward_proxy), which is an intermediary for its associated clients to contact any server, a reverse proxy is an intermediary for its associated servers to be contacted by any client.

Nginx is used as a reverse proxy server in our project.

Functionalities of Nginx in this project:

* Reverse proxies can hide the existence and characteristics of an [origin server](https://en.wikipedia.org/wiki/Origin_server) or servers.
* [Application firewall](https://en.wikipedia.org/wiki/Application_firewall) features can protect against common web-based attacks, like DoS or DDoS

Steps for installing ,configuring and running nginx on port 80 on Linux:

Installation: sudo apt-get install ngnix

Restart nginx: Navigate to /etc/nginx folder

sudo service nginx restart

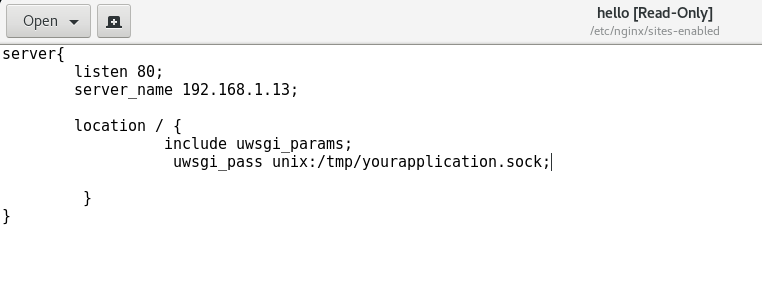
Configuring Nginx for Flask Framework

Installing uWSGI (GateWay Interface for FLASK)

sudo apt-get install build-essential python python-dev

pip install uwsgi

Configuring Nginx

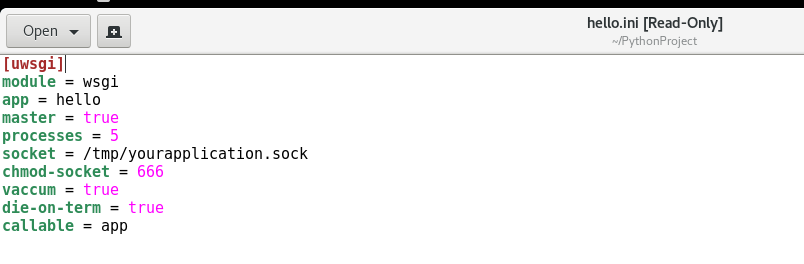


Configuration file for nginx. Configures the server to listen to port 80.

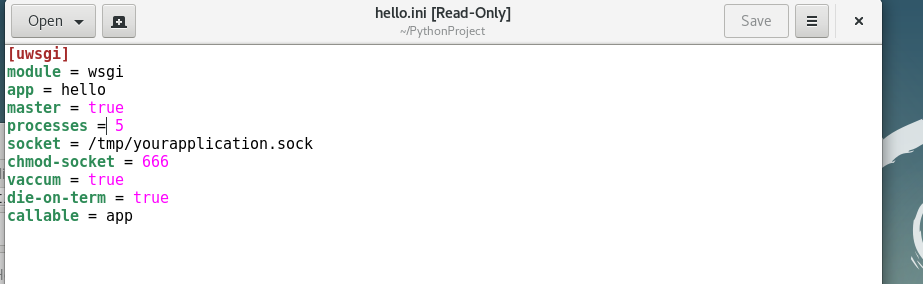
Server\_name : is domain name or ip

Location: socket path provided in the uwsgi ini file.

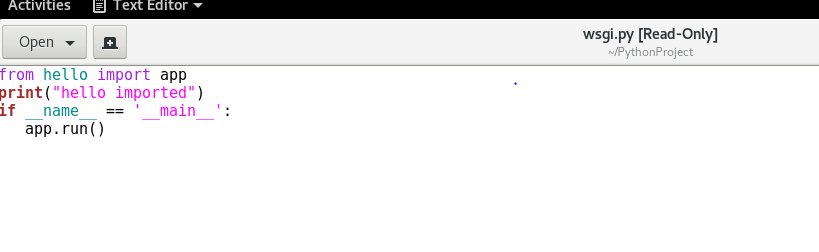
Configuring uWSGI



uWSGI has a socket which listens to communications from nginx and pass it to flask



wsgi hosting the FLASK application

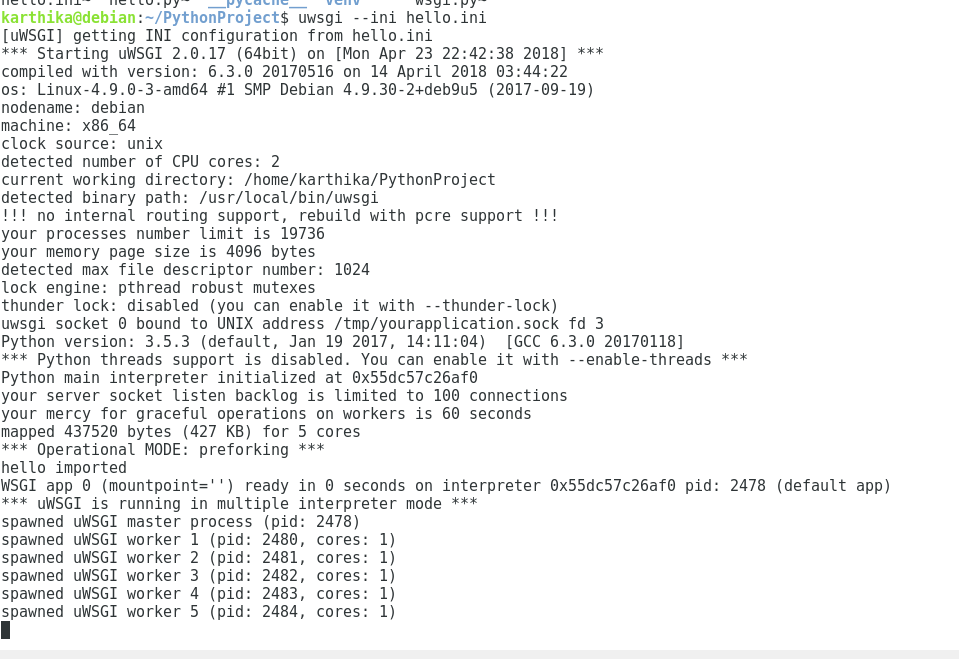


Run the uswgi.ini, so that the gate way is up and running

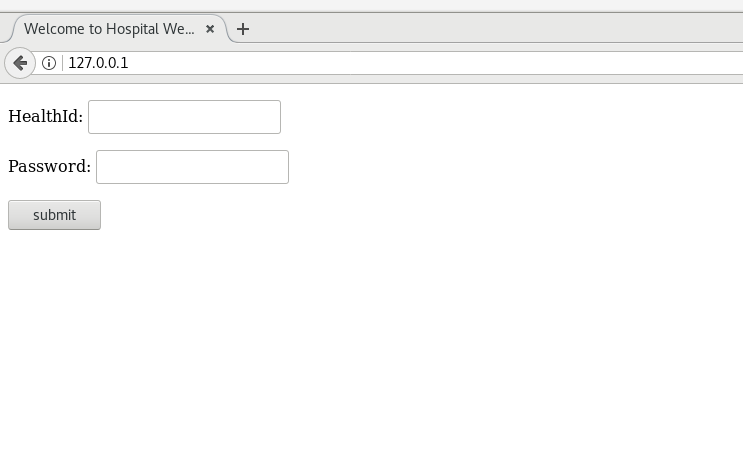
Command for running the .ini file of uwsgi

uwsgi --ini hello.ini

uwsgi is up and running. Since number of processes are given as 5 in the ini file, uwsgi has created 5 processes to handle the requests from nginx



After starting uwsgi, go to the browser and give the ip address. Port 80 is now hosting the hospital website.



# **WINSOCK Implementation**

Client communication with webserver implemented via GET and POST requests. Source code provided in the Appendix.

GCC compiler on windows: Mingw-w64

Command to compile: gcc client.c -o client.exe -lws2\_32 -lmswsock -ladvapi32

Run: client.exe ip\_address\_of\_Server

WSAStartUp:

The **WSAStartup** function initiates use of the Winsock DLL by a process.

Syntax:

int WSAStartup(

\_In\_  WORD      wVersionRequested,

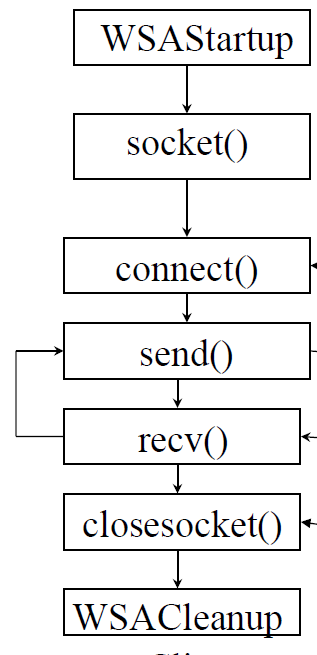
\_Out\_ LPWSADATA lpWSAData

);

wVersionRequested: The highest version of Windows Sockets specification that the caller can use

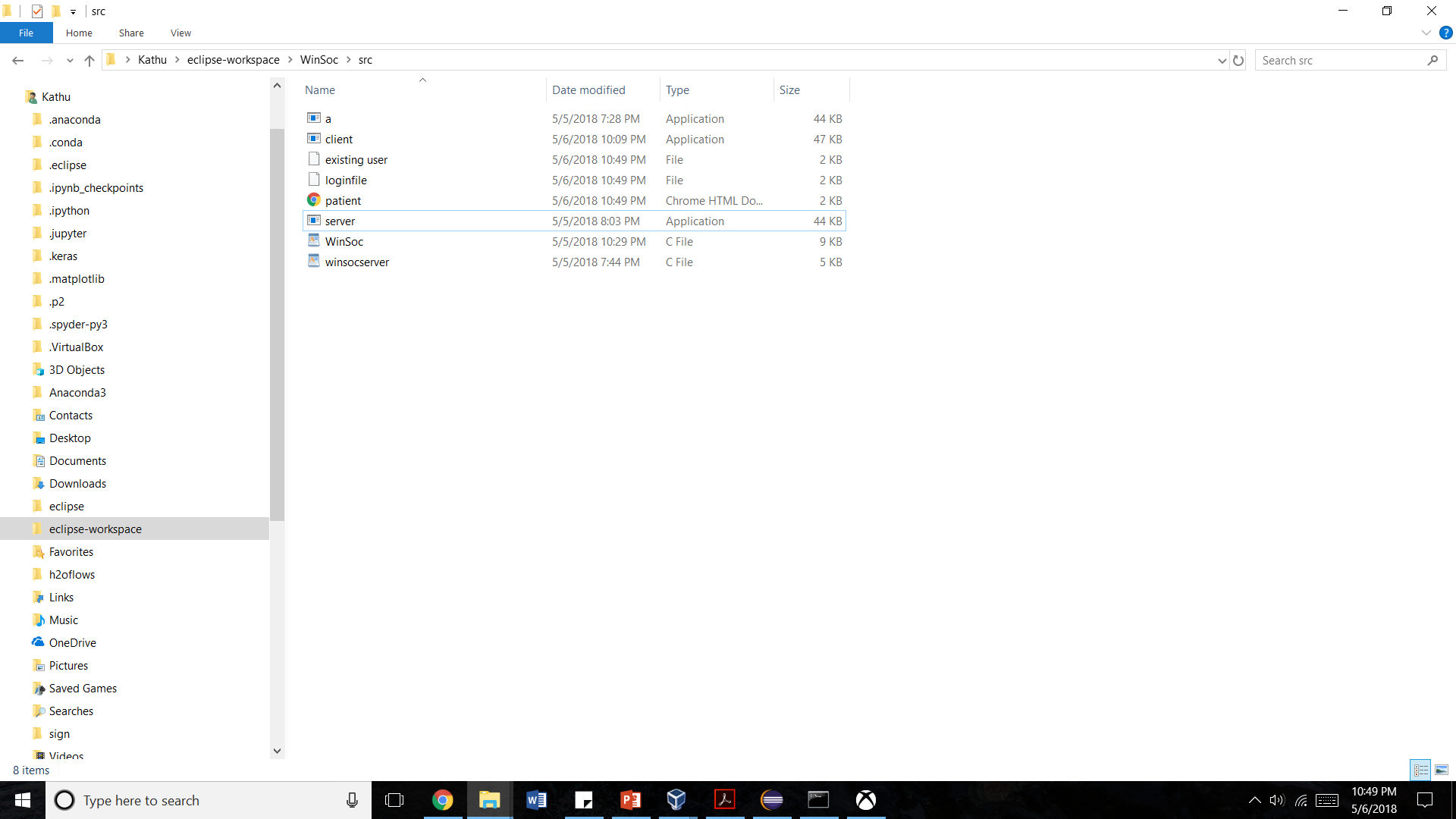
lpWSAData: A pointer to the [**WSADATA**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms741563(v=vs.85).aspx) data structure that is to receive details of the Windows Sockets implementation.

(Reference: https://msdn.microsoft.com/enus/library/windows/desktop/ms742213(v=vs.85).aspx)



WSACleanUp

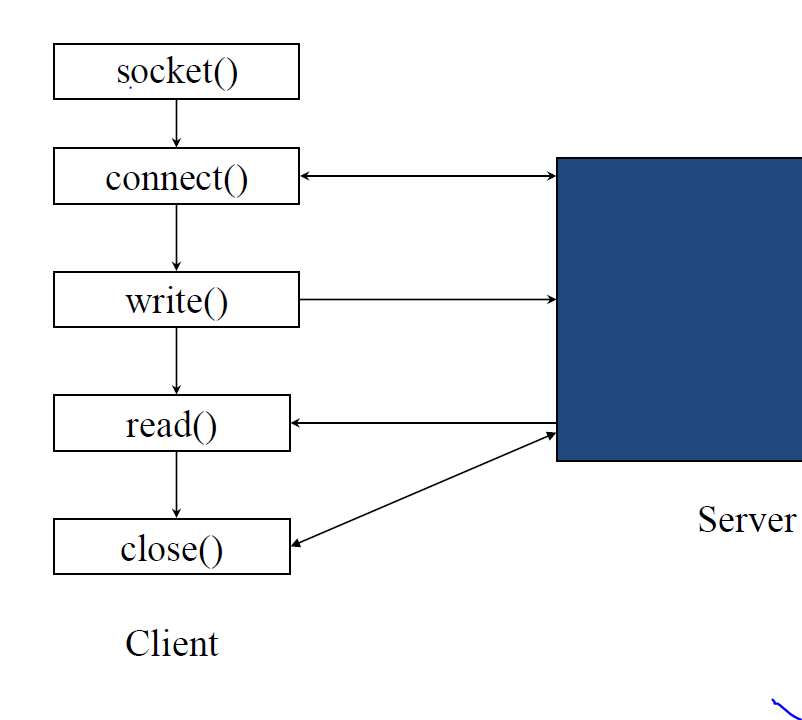
The **WSACleanup** function terminates use of the Winsock 2 DLL (Ws2\_32.dll).



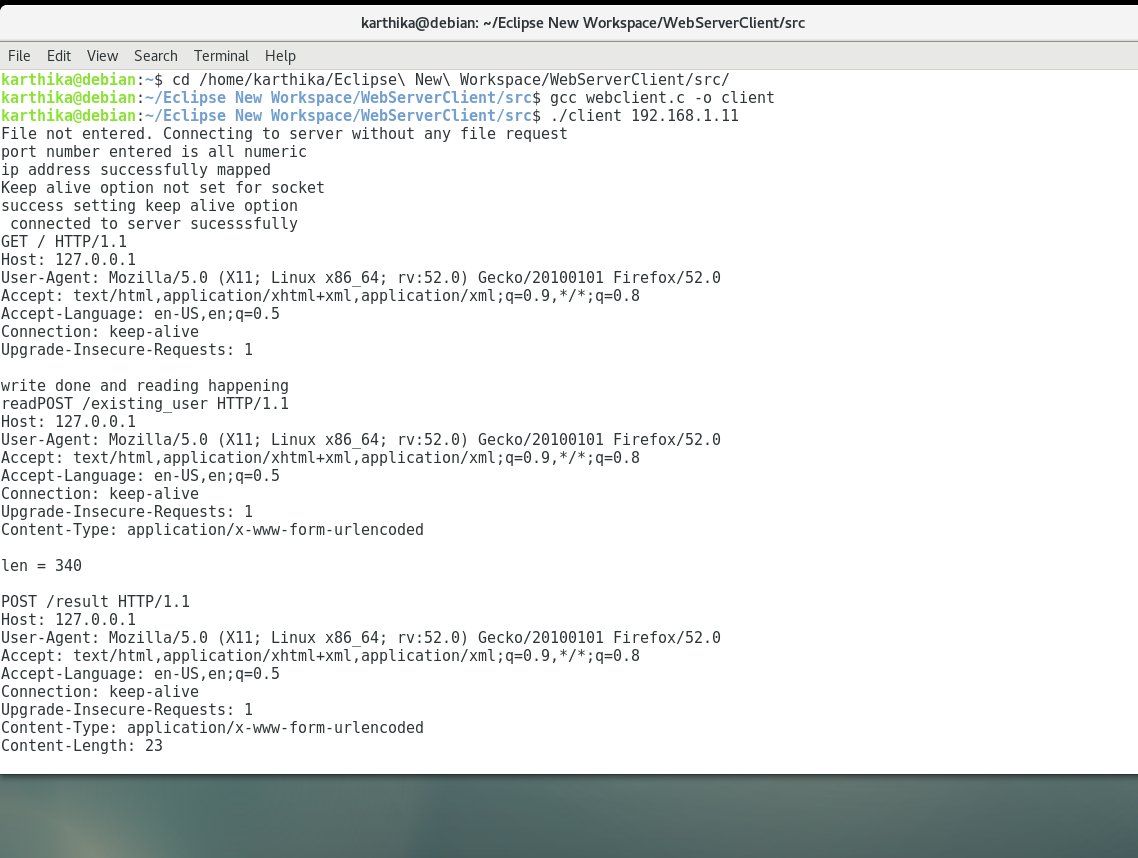
# 

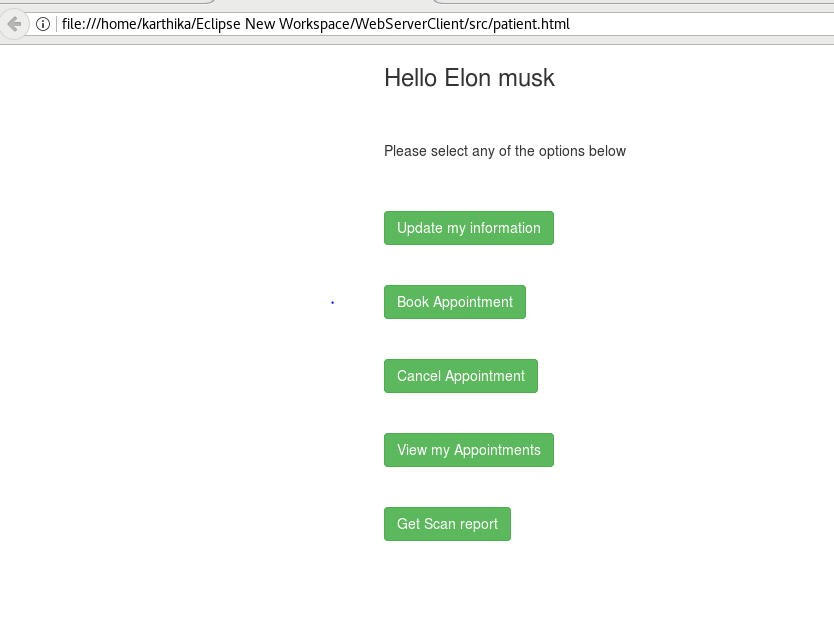
# Client in BSD Sockets

Client communication with webserver implemented via GET and POST requests. Source code provided in the Appendix.



Output:





# 3.Results and analysis

# 

# Fig. The main page that a client is greeted with.

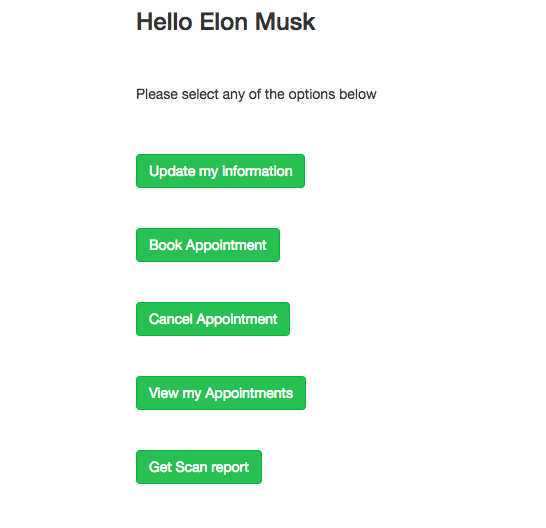
# 

# Fig. The new user page where a new patient can create a user account.

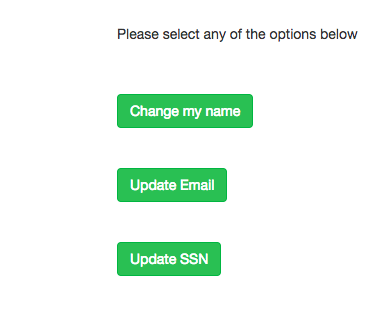
# 

# Fig. The page where different players can login. An invalid credential makes the website reroute to the new user/existing user page

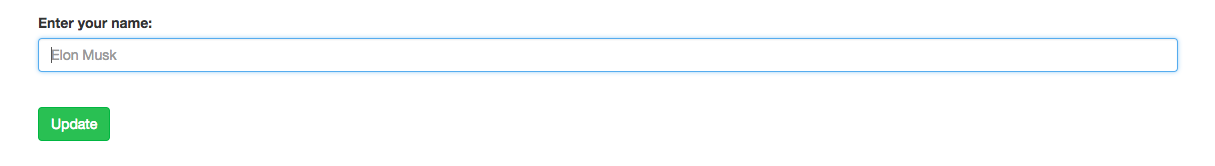
1. **Patient Profile**



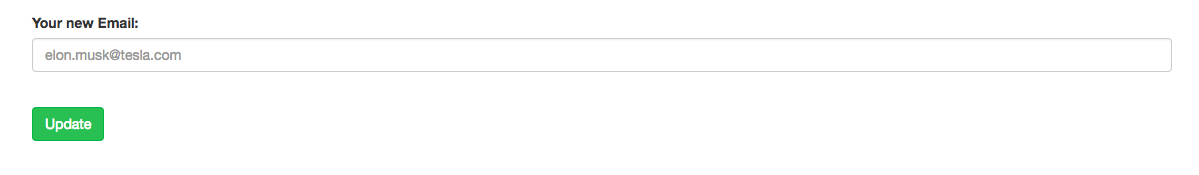
# Fig. The successful login page (result) for a patient named Elon Musk.



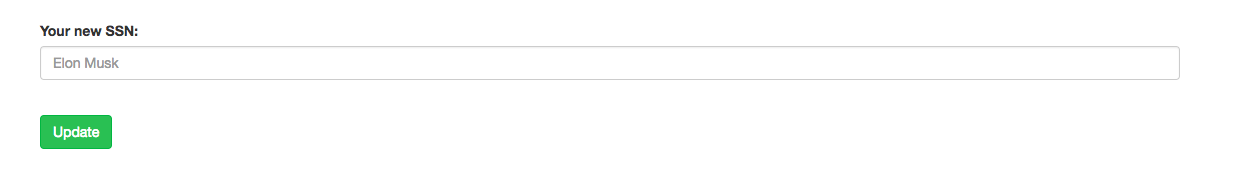
# Fig. The update\_patient\_info page where a patient can update his name, email or SSN.



# Fig. The update\_patient\_name page where a patient can update his name.



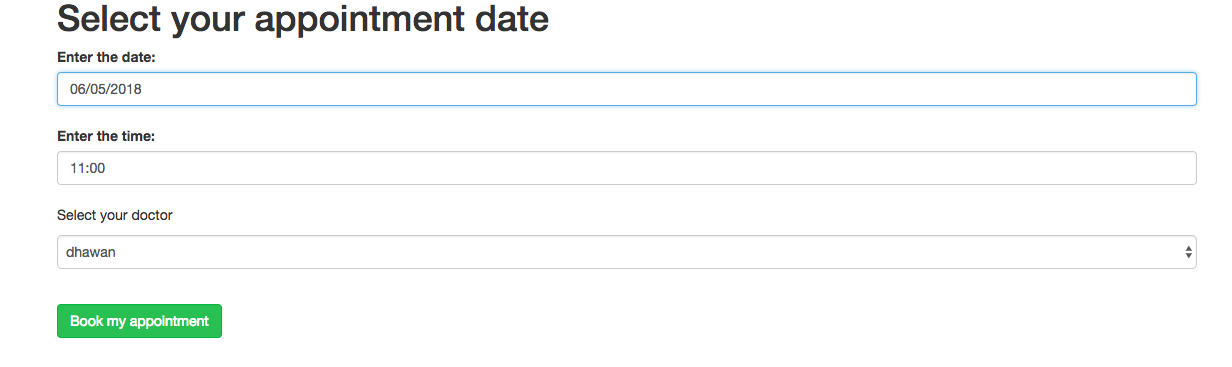
# Fig. The update\_patient\_email page where a patient can update his email.



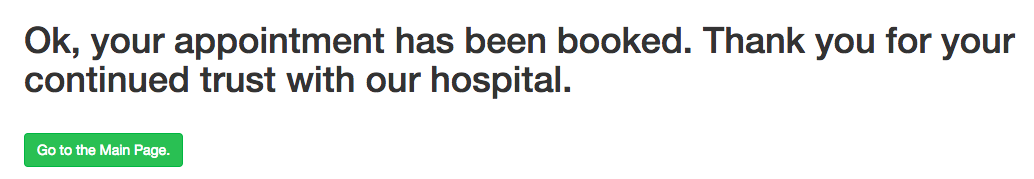
# Fig. The update\_patient\_ssn page where a patient can update his SSN.



# Fig. The update\_patient\_ssn\_success page which gets displayed when a patient has updated his SSN. Similar pages get displayed when a patient updates his name or email.



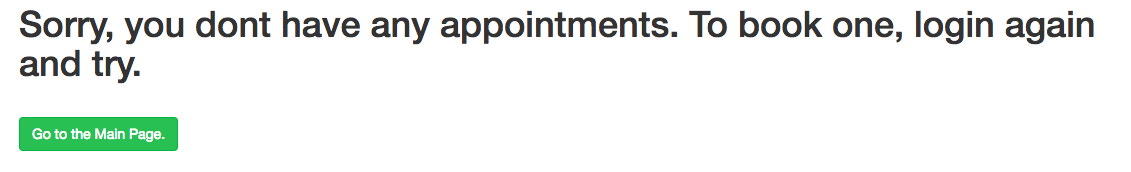
# Fig. The book\_appointment page where a patient can book an appointment.



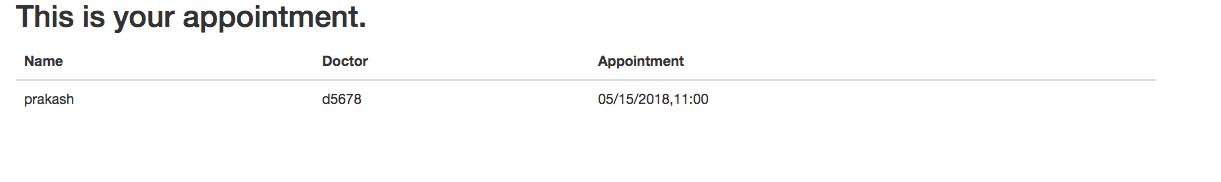
# Fig. The book\_appointment\_success page indicating a successful appointment booking.



Fig. The cancel\_appointment page indicating a successful appointment cancellation.



# Fig. The cancel\_appointment page indicating when no appointment have been previously booked.



# Fig. The view\_appointment page showing all the appointment details for the patient.

A screenshot of a social media post

Description generated with very high confidence

Fig 8: Get scan report-Email sent to the patient’s mail id

1. **Doctor Profile**

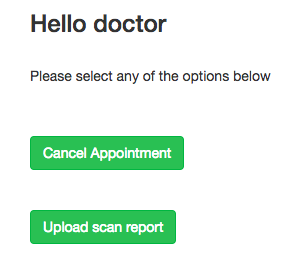


Fig. The login page for a doctor.



# Fig. The cancel\_appointment\_doctor page indicating the successful cancellation of an appointment by the doctor.

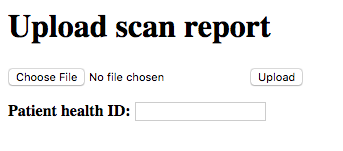
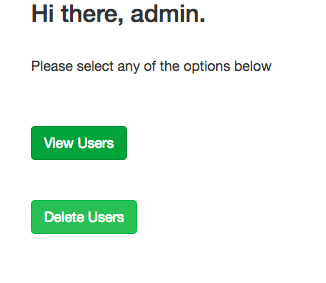


Fig. The uploadFile page where the doctor can upload a patient’s scan report.

1. **Admin Profile**



# Fig. The login page for a admin.

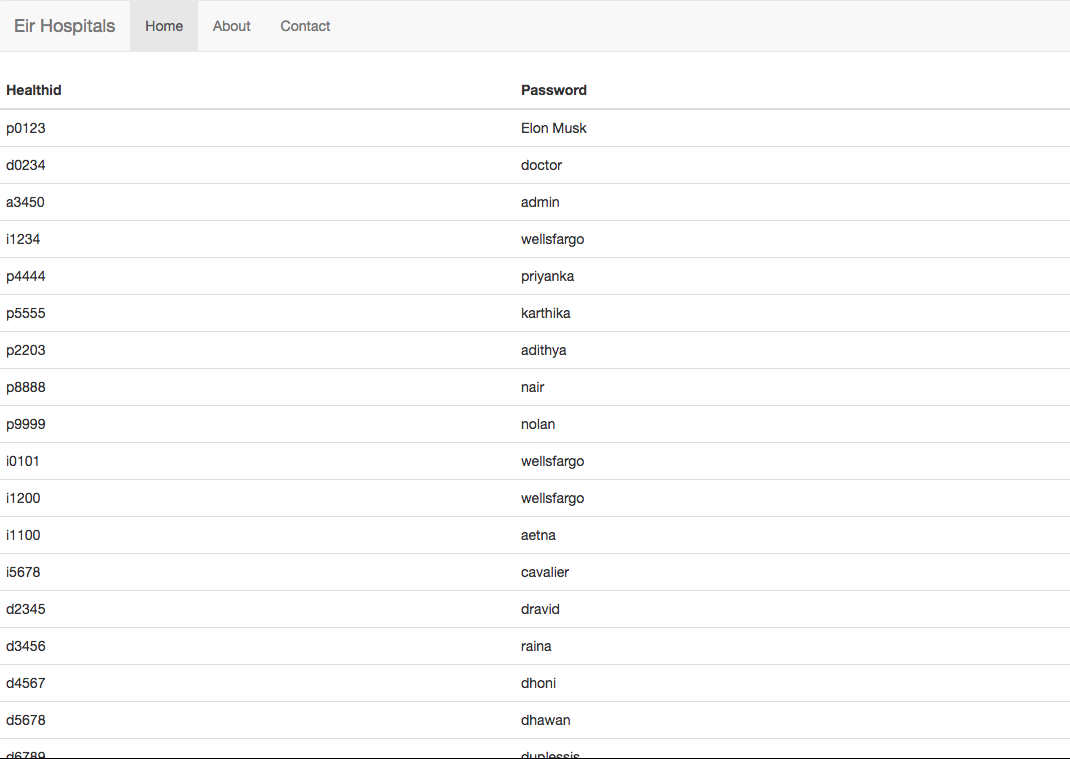
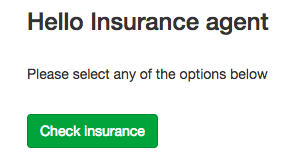


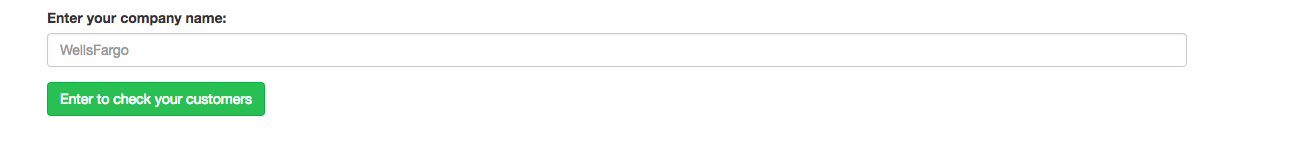
Fig. The admin page where the admin can view all the people involved with the hospital be it patient, doctor, insurance agent or another admin.

# Fig. The admin\_delete page where the admin can erase any account from the database records.

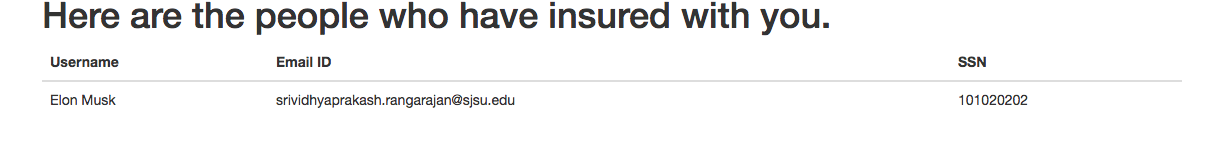
1. **Insurance Agent Profile**



# Fig. The login page for any insurance company.

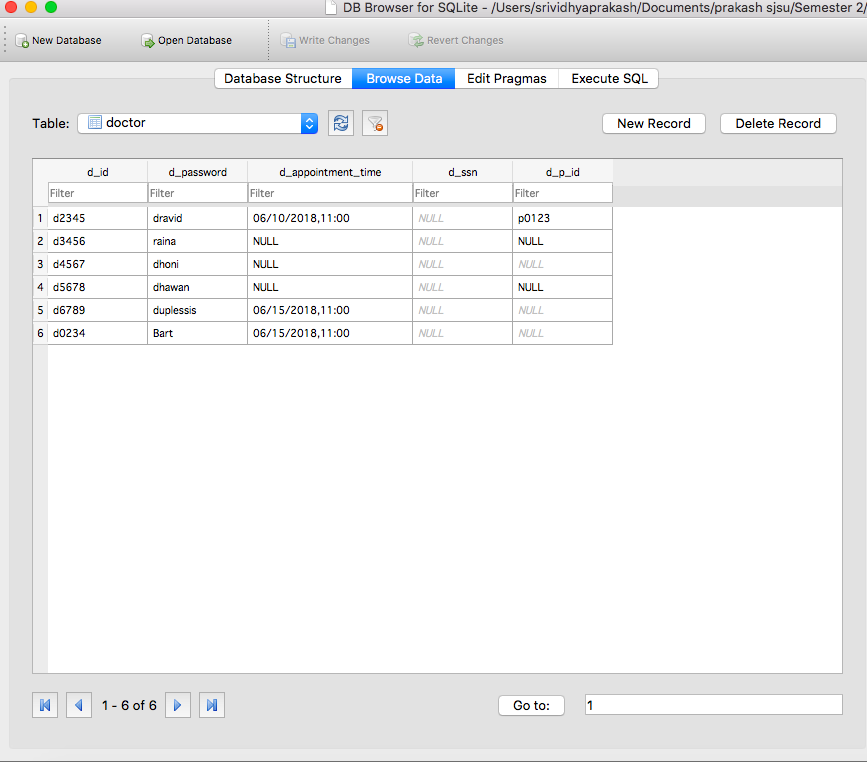


# Fig. The check\_insurance page.

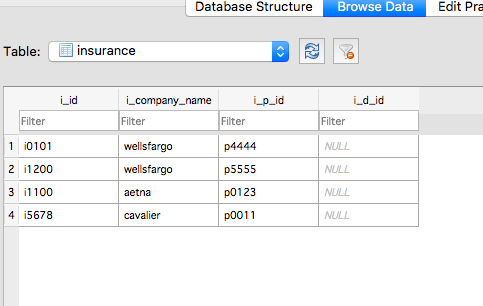


# Fig. The check\_insurance\_result page which shows who all have registered with the insurance company (in this case, aetna)

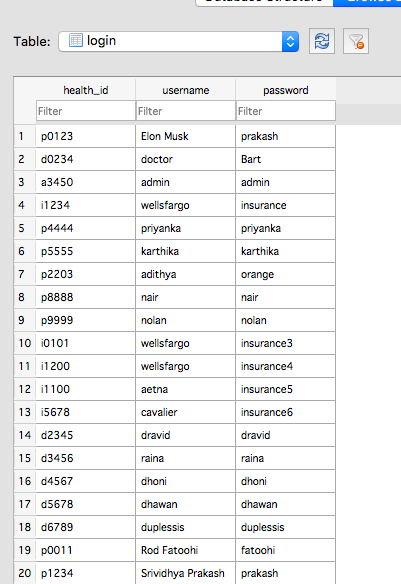
1. **Database**



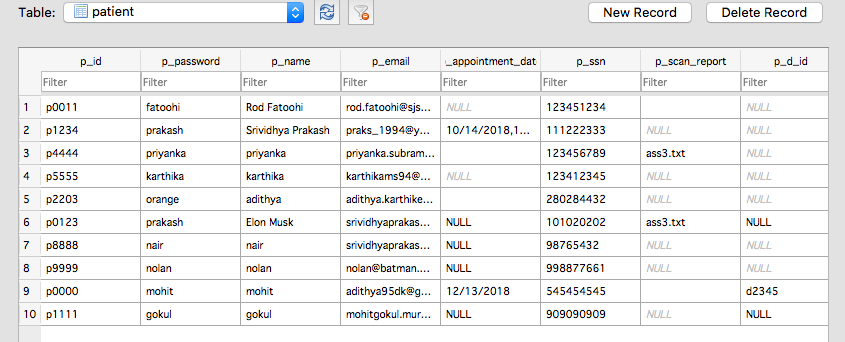
# Fig. The doctor database table.



# Fig. The insurance database table.



# Fig. The login database table.



# Fig. The patient database table.

# 4.Future work

# The project could be easily extended to include a confirmation e-mail when an appointment is made and when an appointment is cancelled. This can be done for both the patient and the doctor. The healthcare system can also be hosted using multiple servers to make use of the load balancing feature by registering a domain name and hosting Nginx with the registered domain name. The front-end for the booking of appointments could be better polished with the use of Google Calendar API. The website in general could use more refined front-end frameworks like Bootstrap to get a better look and feel.

# **5.Conclusion**

This distributed healthcare system thus handles different user personas and their functionalities in a **secure and stable manner**. Using Nginx Reverse Proxy provides security whereas usage of Winsock, BSD Sockets has helped scale the system to include multiple types of users. The usage of Flask framework as the webserver and SqlLite3 as the database has made it simpler to host the application to different clients.

# **6.References**

1. <https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-ii-templates>
2. <http://flask.pocoo.org/>
3. <https://www.tutorialspoint.com/flask/flask_sqlite.htm>
4. <https://stackoverflow.com/>
5. <http://www.leeladharan.com/installing-nginx-flask-uwsgi-on-mac-osx-yosemite>
6. <https://www.w3schools.com/sql/sql_update.asp>
7. <https://stackoverflow.com/questions/46468497/type-of-the-received-date-input-from-html-is-class-string>
8. <https://webcache.googleusercontent.com/search?q=cache:XLFSphmzNrUJ:https://pythonspot.com/flask-web-forms/+&cd=2&hl=en&ct=clnk&gl=us>
9. <http://www.pythonforbeginners.com/cheatsheet/python-file-handling>

# **7.Contribution**

* Priyanka Subramanyam: Contribution to front end creation and validation, Nginx Reverse Proxy.
* Karthika Madhavanpillai Sasidharan Nair: Contribution to connection between front end and client server model, Winsock.
* SriVidhya Prakash Rangarajan: Contribution to Server – DB connection, BSD Sockets.

# **8.APPENDICES**

1. **CODE**

Hello.ini

[uwsgi]

module = wsgi

app = hello

master = true

processes = 5

socket = /tmp/yourapplication.sock

chmod-socket = 666

vaccum = true

die-on-term = true

callable = app

wsgi.py

from hello import app

print("hello imported")

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

Hello.py

from flask import Flask, render\_template, request, redirect, url\_for, flash

import sqlite3 as sql

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.base import MIMEBase

from email import encoders

import datetime

import os

import tablib

from werkzeug.utils import secure\_filename

from flask import send\_from\_directory

from flask\_session import Session

UPLOAD\_FOLDER = '/Users/srividhyaprakash/Documents/flask\_uploads/'

ALLOWED\_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'])

app = Flask(\_\_name\_\_, template\_folder='template')

app.secret\_key = "super secret key"

app.config['SESSION\_TYPE'] = 'filesystem'

sess = Session()

patient\_id = "global\_variable"

user="global user"

@app.route('/')

def main\_page():

print("entered here")

return render\_template('main\_page.html')

@app.route('/new\_user',methods = ['POST', 'GET'])

def new\_user():

if (request.method == 'POST'):

return render\_template("new\_user.html")

@app.route('/insert\_patient\_data',methods = ['POST', 'GET'])

def insert\_patient\_data():

if request.method == 'POST':

try:

conn = sql.connect('db\_1.0.db')

print("Opened database successfully")

p\_id = request.form['id\_of\_user'] # have username field as uname in html code

ssn = request.form['ssn\_of\_user'] # have ssn field as myssn in html code

password = request.form['psw\_of\_user'] # have password field as pswd in html code

name = request.form['name\_of\_user'] # have password field as pswd in html code

email\_id = request.form['email\_of\_user'] # have password field as pswd in html code

print("id Value properly received", p\_id)

cur = conn.cursor()

cur.execute("INSERT INTO patient (p\_id,p\_password,p\_name,p\_email,p\_ssn) VALUES (?,?,?,?,?)",(p\_id,password,name,email\_id,ssn) )

cur.execute("INSERT INTO login (health\_id,username, password) VALUES (?,?,?)", (p\_id,name, password))

conn.commit()

print("successfully inserted value in database")

msg = "Your record has been updated in the database."

except:

print("error in database connection or access")

msg = "error in insert operation"

conn.rollback()

finally:

print("closing connection")

conn.close()

return render\_template('new\_user\_success.html', msg = msg) # go back to patient.html and render with the msg block dynamically

@app.route('/existing\_user',methods = ['POST', 'GET'])

def existing\_user():

if (request.method == 'POST'):

return render\_template("input.html")

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

def success():

return redirect(url\_for('main\_page'))

@app.route('/book\_appointment',methods = ['POST', 'GET'])

def book\_appointment():

if (request.method == 'POST'):

try:

conn = sql.connect('db\_1.0.db')

cur = conn.cursor()

cur.execute("select \* from doctor")

rows = cur.fetchall()

finally:

return render\_template("appointment\_date\_fix.html", result = rows)

@app.route('/book\_appointment\_success',methods = ['POST', 'GET'])

def book\_appointment\_success():

if (request.method == 'POST'):

try:

conn = sql.connect('db\_1.0.db')

cur = conn.cursor()

print("Opened database successfully")

doctor\_id = request.form["d\_id"]

print("the doctor\_id is ", doctor\_id)

print("the type of doctor id is ", type(doctor\_id))

the\_date = request.form["appointment\_date"]

the\_time = request.form["appointment\_time"]

print("the date is %s" % the\_date)

print("the time is %s" % the\_time)

valid\_date = the\_date.split('/', 2)

print("the date after splitting is ", the\_date)

valid\_time = the\_time.split(':', 1)

print("the time after splitting is", the\_time)

if(len(valid\_date) == 3 and len(valid\_time) == 2 ):

the\_date = the\_date + "," + the\_time

print("the date inside the if condition is ", the\_date)

print("the patient\_id is ", patient\_id)

print("the type of patient\_id is ", type(patient\_id))

print(" the statement is UPDATE doctor SET d\_appointment\_time = %s, d\_p\_id = %s where d\_id = %s" % (the\_date, patient\_id, doctor\_id))

cur.execute("UPDATE doctor SET d\_appointment\_time = ?, d\_p\_id = ? where d\_id = ?", (the\_date, patient\_id, doctor\_id))

cur.execute("UPDATE patient SET p\_appointment\_date = ?, p\_d\_id = ? where p\_id = ?",(the\_date, doctor\_id, patient\_id))

print("executed the command")

conn.commit()

msg = "Ok, your appointment has been booked. Thank you for your continued trust with our hospital."

else:

msg = "Please enter a valid date and time"

# print("got p\_id", p\_id)

# cur.execute("select \* from login")

# cur.execute("UPDATE patient SET p\_appointment\_date = ? where p\_id = ?",('NULL', patient\_id))

# print("executed the command")

# conn.commit()

except:

msg = "Sorry, your appointment could not be booked. Please login and try again."

finally:

print("closing database connection")

conn.close()

return render\_template("scan\_report\_success.html", msg = msg)

@app.route('/cancel\_appointment',methods = ['POST', 'GET'])

def cancel\_appointment():

if (request.method == 'POST'):

try:

conn = sql.connect('db\_1.0.db')

cur = conn.cursor()

# cur.execute("select \* from login")

cur.execute("select p\_appointment\_date, p\_d\_id from patient where p\_id= ?",(patient\_id,))

value=cur.fetchall()

print("Value for appointment is ", value)

doctor\_id = value[0][1]

if(value[0][0]== 'NULL'):

msg="Sorry, you dont have any appointments. To book one, login again and try."

else:

cur.execute("UPDATE patient SET p\_appointment\_date = ?, p\_d\_id = ? where p\_id = ?",('NULL', 'NULL', patient\_id))

cur.execute("UPDATE doctor SET d\_appointment\_time = ?, d\_p\_id = ? where d\_id = ?", ('NULL', 'NULL', doctor\_id))

print("executed the command")

conn.commit()

msg = "Ok, your appointment has been cancelled. Please login to book another appointment."

except:

msg = "Sorry, your appointment could not be cancelled. Please login and try again."

finally:

print("closing database connection")

conn.close()

return render\_template("scan\_report\_success.html", msg = msg)

@app.route('/view\_appointment',methods = ['POST', 'GET'])

def view\_appointment():

if (request.method == 'POST'):

try:

global patient\_id

conn = sql.connect('db\_1.0.db')

cur = conn.cursor()

cur.execute("select \* from patient where p\_id= ?",(patient\_id,))

rows = cur.fetchall()

print("patient\_id is: %s", patient\_id)

print(rows)

finally:

print("closing database connection")

conn.close()

return render\_template("view\_appointment.html", result = rows)

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

def result():

if (request.method == 'POST'):

try:

conn = sql.connect('db\_1.0.db')

print("Opened database successfully")

username = request.form['uname']

global patient\_id

patient\_id = username

print("Username is: %s" % patient\_id)

password = request.form['psw']

print("password is: %s" % password)

dict\_login = {'patient\_id':patient\_id, 'password' : password}

print("created the dictionary")

cur = conn.cursor()

print("created the cursor")

cur.execute("select \* from login")

print("executed the query")

rows = cur.fetchall()

print("the type of rows is", type(rows))

print(rows)

for row in rows:

print("the row in consideration is", row)

if row[0] == patient\_id and row[2] == password:

print("patient\_id present in the database")

if(row[0][0] == 'p'):

return render\_template("patient.html", result = row)

elif (row[0][0] == 'd'):

return render\_template("doctor.html", result = row)

elif (row[0][0] == 'i'):

return render\_template("insurance.html", result = row)

elif (row[0][0] == 'a'):

return render\_template("admin\_main.html", result = row)

print("Username incorrect")

return redirect(url\_for('main\_page'))

except:

print("error in database connection or access")

finally:

print("closing database connection")

conn.close()

@app.route('/check\_insurance',methods = ['POST', 'GET'])

def check\_insurance():

print("Entered the check insurance flask function")

if (request.method == 'POST'):

return render\_template("check\_insurance.html")

@app.route('/check\_insurance\_result',methods = ['POST', 'GET'])

def check\_insurance\_result():

print("Entered the check insurance result flask function")

if request.method == 'POST':

try:

conn = sql.connect('db\_1.0.db')

print("Opened database successfully")

i\_name = request.form['insurance\_comp\_name']

print("company name received is", i\_name)

cur = conn.cursor()

cur.execute("select patient.p\_name, patient.p\_email, patient.p\_ssn from patient, insurance where insurance.i\_company\_name = ? and insurance.i\_p\_id = patient.p\_id", (i\_name,))

print("executed the function successfully")

rows = cur.fetchall()

print("fetched all rows")

print("length of rows is", len(rows))

print("type of rows is", type(rows))

print("type of rows is", rows)

# check the type here and print accordingly or append to a list and send that to check\_insurance\_result.html

except:

print("error in database connection or access")

finally:

print("closing database connection")

conn.close()

return render\_template("check\_insurance\_result.html", result = rows)

@app.route('/scan\_report',methods = ['POST', 'GET'])

def scan\_report():

if(request.method == 'POST'):

try:

print("Python code to send an email should go here")

fromaddr = "srividhyaprakash029@gmail.com"

conn = sql.connect('db\_1.0.db')

print("Opened database successfully")

cur = conn.cursor()

cur.execute("select patient.p\_email, patient.p\_scan\_report from patient where patient.p\_id = ?", (patient\_id,))

returned\_list = cur.fetchall()

print("The email address (toaddr) should come here", returned\_list)

print("the type of toaddr is ", type(returned\_list))

print(returned\_list[0][0])

toaddr = returned\_list[0][0]

# filename = "d4ecb063-7ece-40b5-b92b-a3a1f04ada34-original.jpeg"

filename = returned\_list[0][1]

print("just the filename fetched from the server is ", filename)

body = "Your scan report has been attached along with this mail. \

Hope you liked your stay at our hospital."

msg = MIMEMultipart()

msg['From'] = fromaddr

msg['To'] = toaddr

msg['Subject'] = "The hospital wishes you a speedy recovery."

full\_path\_of\_file = UPLOAD\_FOLDER + filename

print("the full path of the file is", full\_path\_of\_file)

msg.attach(MIMEText(body, 'plain'))

attachment = open(full\_path\_of\_file, 'rb')

msg\_send = "Your scan report has been successfully sent to you!"

part = MIMEBase('application', 'octet-stream')

part.set\_payload((attachment).read())

encoders.encode\_base64(part)

part.add\_header('Content-Disposition', "attachment;filename=%s" % filename)

msg.attach(part)

# 'your host address', 'your port number'

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

# security function used to protect your password

server.starttls()

server.login(fromaddr, "007029svp1") # password should be changed here

msg = msg.as\_string()

# "my email", "email address to send to", "msg"

server.sendmail(fromaddr, toaddr, msg)

server.quit()

except:

msg\_send = "Sorry, your scan report was not found in the database"

finally:

print("closing database connection")

conn.close()

return render\_template("scan\_report\_success.html", msg = msg\_send)

@app.route('/update\_patient\_info', methods = ['GET', 'POST'])

def update\_patient\_info():

return render\_template('update\_patient\_info\_page.html')

@app.route('/update\_patient\_email',methods = ['POST', 'GET'])

def update\_patient\_email():

return render\_template('update\_patient\_email.html')

@app.route('/update\_patient\_email\_success',methods = ['POST', 'GET'])

def update\_patient\_email\_success():

if request.method == 'POST':

try:

conn = sql.connect('db\_1.0.db')

print("Opened database successfully")

# p\_id = request.form["id\_of\_user"]

email = request.form["email\_of\_user"]

cur = conn.cursor()

print("created the cursor")

cur.execute("UPDATE patient SET p\_email = ? where p\_id = ?",(email,patient\_id))

conn.commit()

msg = "Your Email is successfully updated."

except:

msg = "Error in updating your Email, login and try again."

finally:

return render\_template("scan\_report\_success.html", msg = msg)

@app.route('/update\_patient\_ssn',methods = ['POST', 'GET'])

def update\_patient\_ssn():

return render\_template('update\_patient\_ssn.html')

@app.route('/update\_patient\_ssn\_success',methods = ['POST', 'GET'])

def update\_patient\_ssn\_success():

if request.method == 'POST':

try:

conn = sql.connect('db\_1.0.db')

print("Opened database successfully")

# p\_id = request.form["id\_of\_user"]

ssn = request.form["ssn\_of\_user"]

cur = conn.cursor()

print("created the cursor")

cur.execute("UPDATE patient SET p\_ssn = ? where p\_id = ?",(ssn,patient\_id))

conn.commit()

msg = "Your SSN is successfully updated."

except:

msg = "Error in updating SSN, login and try again."

finally:

return render\_template("scan\_report\_success.html", msg = msg)

@app.route('/update\_patient\_name',methods = ['POST', 'GET'])

def update\_patient\_first\_name():

return render\_template('update\_patient\_name.html')

@app.route('/update\_patient\_name\_success',methods = ['POST', 'GET'])

def update\_patient\_name\_success():

if request.method == 'POST':

try:

conn = sql.connect('db\_1.0.db')

print("Opened database successfully")

# p\_id = request.form["id\_of\_user"]

new\_name = request.form["name\_of\_user"]

cur = conn.cursor()

print("created the cursor")

cur.execute("UPDATE patient SET p\_name = ? where p\_id = ?",(new\_name,patient\_id))

cur.execute("UPDATE login SET username = ? where health\_id = ?",(new\_name,patient\_id))

conn.commit()

msg = "Your name has been successfully updated."

except:

msg = "Error in updating your name, login and try again."

finally:

return render\_template("scan\_report\_success.html", msg = msg)

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

def allowed\_file(filename):

return '.' in filename and \

filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

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

def upload\_file():

if request.method == 'POST':

# pass

# check if the post request has the file part

try:

print("Entering upload\_file")

if 'file' not in request.files:

flash('No file part')

return redirect(request.url)

file = request.files['file']

print("printing file", file)

print("printing file.filename", file.filename)

print("type of file", type(file))

scan\_report\_patient\_id = request.form["patient\_id\_file\_upload"]

# if user does not select file, browser also

# submit a empty part without filename

if file.filename == '':

flash('No selected file')

return redirect(request.url)

if file and allowed\_file(file.filename):

filename = secure\_filename(file.filename)

file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))

conn = sql.connect('db\_1.0.db')

print("Opened database successfully")

cur = conn.cursor()

print("created the cursor")

cur.execute("UPDATE patient set p\_scan\_report = ? where p\_id = ?", (file.filename, scan\_report\_patient\_id))

conn.commit()

print("closing database connection")

conn.close()

return redirect(url\_for('uploaded\_file'))

except:

print("Entered the except block in the upload\_file function")

return render\_template("uploadFile.html")

# pass

return render\_template("uploadFile.html")

@app.route('/uploaded\_file')

def uploaded\_file():

print("entering to go back to the main page")

return redirect(url\_for("main\_page"))

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

def admin():

if (request.method == 'POST'):

try:

conn = sql.connect('db\_1.0.db')

print("Opened database successfully")

cur = conn.cursor()

print("created the cursor")

cur.execute("select \* from login")

rows = cur.fetchall()

print(rows)

finally:

print("closing database connection")

conn.close()

return render\_template("admin.html", result = rows)

@app.route('/admin\_delete',methods = ['POST', 'GET'])

def admin\_delete():

# if (request.method == 'POST'):

if (request.method == 'POST'):

try:

# global user

conn = sql.connect('db\_1.0.db')

print("Opened database successfully")

cur = conn.cursor()

print("created the cursor")

cur.execute("select \* from login")

rows = cur.fetchall()

# user = request.form["selecteduser"]

# print("user:",user)

print(rows)

finally:

print("closing database connection")

conn.close()

return render\_template("admin\_delete.html", result = rows)

@app.route('/cancel\_appointment\_doctor',methods = ['POST', 'GET'])

def cancel\_appointment\_doctor():

if (request.method == 'POST'):

try:

conn = sql.connect('db\_1.0.db')

cur = conn.cursor()

# cur.execute("select \* from login")

cur.execute("select d\_appointment\_time, d\_p\_id from doctor where d\_id= ?",(patient\_id,))

value=cur.fetchall()

print("Value for appointment is ", value)

patient\_id\_in\_doctor = value[0][1]

if(value[0][0]== 'NULL'):

msg="Sorry doctor, you dont have any appointments."

else:

print("patient\_id\_in\_doctor is ", patient\_id\_in\_doctor)

print(" the type of patient\_id\_in\_doctor is ", type(patient\_id\_in\_doctor))

print("the doctor id which is nothing but patient\_id here is ", patient\_id)

print("the type of doctor id which is nothing but patient\_id here is ", type(patient\_id))

print("the statement is UPDATE doctor SET d\_appointment\_time = %s, d\_p\_id = %s where d\_id = %s " % ('NULL', 'NULL', patient\_id))

cur.execute("UPDATE doctor SET d\_appointment\_time = ?, d\_p\_id = ? where d\_id = ?", ('NULL', 'NULL', patient\_id))

cur.execute("UPDATE patient SET p\_appointment\_date = ?, p\_d\_id = ? where p\_id = ?",('NULL', 'NULL', patient\_id\_in\_doctor))

print("executed the command")

conn.commit()

msg = "Ok doctor, your appointment has been cancelled."

except:

msg = "Sorry doctor, unknown error occured. Please login and try again."

finally:

print("closing database connection")

conn.close()

return render\_template("scan\_report\_success.html", msg = msg)

@app.route('/admin\_delete\_success',methods = ['POST', 'GET'])

def admin\_delete\_success():

if (request.method == 'POST'):

try:

global user

conn = sql.connect('db\_1.0.db')

print("Opened database successfully")

cur = conn.cursor()

user = request.form["selecteduser"]

print("user:%s",user)

print("created the cursor")

cur.execute("delete from login where health\_id= ?",(user,))

if(user[0] == 'p'):

cur.execute("delete from patient where p\_id= ?",(user,))

elif (user[0] == 'd'):

cur.execute("delete from doctor where d\_id= ?",(user,))

elif (user[0] == 'i'):

cur.execute("delete from insurance where i\_id= ?",(user,))

conn.commit()

msg = "Deleted the account, log back in to delete or view account."

except:

print("Entering the admin delete success except block")

msg = "Sorry, either the account doesn't exist or some other error occured.\n Log back in to try again."

finally:

print("closing database connection")

conn.close()

return render\_template("scan\_report\_success.html", msg = msg)

if \_\_name\_\_ == '\_\_main\_\_':

app.secret\_key = 'hdwi891y2erhfiuwela'

app.config['SESSION\_TYPE'] = 'filesystem'

sess.init\_app(app)

app.debug = True

app.run(host = '192.168.1.11', debug = True)

admin.html

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

</head>

<body>

<nav class="navbar navbar-default">

<div class="container-fluid">

<div class="navbar-header">

<a class="navbar-brand" href="#">Eir Hospitals</a>

</div>

<ul class="nav navbar-nav">

<li class="active"><a href="#">Home</a></li>

<li><a href="#/about">About</a></li>

<li><a href="#/contact">Contact</a></li>

</ul>

</div>

</nav>

<div class="table-responsive">

<table class="table">

<thead>

<tr>

<th>Healthid</th>

<th>Password</th>

</tr>

</thead>

<tbody>

{% for row in result %}

<tr>

<td> {{ row[0] }} </td>

<td> {{ row[1] }} </td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</body>

</html>

Admin\_delete.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Appointment</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<h1>Select the account you want to delete</h1>

<form action="{{ 'http://127.0.0.1:5000/admin\_delete\_success' }}" method="POST" role="form">

<div class="form-group">

<br>

<select name="selecteduser" id="selecteduser" class="form-control">

{% for row in result %}

<option value="{{row[0]}}">{{ row[1] }}</option>

{% endfor %}

</select>

</div>

<button type="submit" class="btn btn-success">Delete this account</button>

</form>

<br>

</div>

<br>

</body>

</html>

Admin\_main.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Admin Page</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<h3>Hi there, admin.</h3>

<br>

<p> Please select any of the options below </p>

<br>

<br>

<form action="{{ 'http://127.0.0.1:5000/admin' }}" method="POST" role="form">

<button type="submit" class="btn btn-success">View Users</button>

</form>

<br>

<br>

<form action="{{ 'http://127.0.0.1:5000/admin\_delete' }}" method="POST" role="form">

<button type="submit" class="btn btn-success">Delete Users</button>

</form>

<br>

<br>

</div>

<br>

</body>

</html>

Appointment\_date\_fix.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Appointment</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<h1>Select your appointment date</h1>

<form action="{{ 'http://127.0.0.1:5000/book\_appointment\_success' }}" method="POST" role="form">

<div class="form-group">

<label for="appointment\_date">Enter the date:</label>

<input type="text" class="form-control" name="appointment\_date" required placeholder="06/15/2018">

<br>

<label for="appointment\_time">Enter the time:</label>

<input type="text" class="form-control" name="appointment\_time" required placeholder="11:00">

<br>

<p> Select your doctor </p>

<select name="d\_id" class="form-control">

{% for row in result %}

<option value="{{row[0]}}">{{ row[1] }}</option>

{% endfor %}

</select>

<br>

</div>

<button type="submit" class="btn btn-success">Book my appointment</button>

</form>

<br>

</div>

</body>

</html>

Check\_insurance.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Insurance Company</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<form action="{{ 'http://127.0.0.1:5000/check\_insurance\_result' }}" method="POST" role="form">

<div class="form-group">

<br>

<br>

<label for="insurance\_comp\_name">Enter your company name:</label>

<input type="text" class="form-control" id="insurance\_comp\_name" name="insurance\_comp\_name" placeholder="WellsFargo" required>

</div>

<button type="submit" class="btn btn-success">Enter to check your customers</button>

</form>

<br>

</div>

<br>

</body>

</html>

Check\_insurance\_result.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Customers Page</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<h1>Here are the people who have insured with you.</h1>

<div class="table-responsive">

<table class="table">

<thead>

<tr>

<th>Username</th>

<th>Email ID </th>

<th> SSN</th>

</tr>

</thead>

<tbody>

{% for row in result %}

<tr>

<td> {{ row[0] }} </td>

<td> {{ row[1] }} </td>

<td> {{ row[2] }} </td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

<br>

</div>

<br>

</body>

</html>

Doctor.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Doctor Page</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<h3>Hello doctor {{ name }}</h3>

<br>

<p> Please select any of the options below </p>

<br>

<br>

<form action="{{ 'http://127.0.0.1:5000/cancel\_appointment\_doctor' }}" method="POST" role="form">

<button type="submit" class="btn btn-success">Cancel Appointment</button>

</form>

<br>

<br>

<form action="{{ 'http://127.0.0.1:5000/uploadFile' }}" method="POST" role="form">

<button type="submit" class="btn btn-success">Upload scan report</button>

</form>

<br>

<br>

</div>

<br>

</body>

</html>

Input.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Main Page</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<h1>Welcome to the Official Hospital website</h1>

<form action="{{ 'http://127.0.0.1:5000/result' }}" method="POST" role="form">

<div class="form-group">

<label for="uname">Username:</label>

<input type="text" class="form-control" id="uname" name="uname" placeholder="What's your health ID?">

<br>

<label for="psw">Password:</label>

<input type="password" class="form-control" id="psw" name="psw" placeholder="Enter a password.">

</div>

<button type="submit" class="btn btn-success">Login</button>

</form>

<br>

</div>

<br>

</body>

</html>

Insurance.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Doctor Page</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<h3>Hello Insurance agent {{ name}}</h3>

<br>

<p> Please select any of the options below </p>

<br>

<form action="{{ 'http://127.0.0.1:5000/check\_insurance' }}" method="POST" role="form">

<button type="submit" class="btn btn-success">Check insurance</button>

</form>

<br>

<br>

</div>

<br>

</body>

</html>

Main\_page.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

</head>

<body>

<nav class="navbar navbar-default">

<div class="container-fluid">

<div class="navbar-header">

<a class="navbar-brand" href="#">Eir Hospitals</a>

</div>

<ul class="nav navbar-nav">

<li class="active"><a href="#">Home</a></li>

<li><a href="/about">About</a></li>

<li><a href="/contact">Contact</a></li>

</ul>

</div>

</nav>

<div class="container">

<br>

<br>

<form action="{{ 'http://127.0.0.1:5000/new\_user' }}" method="POST" role="form">

<button type="submit" class="btn btn-success">New User</button>

</form>

<br>

<br>

<form action="{{ 'http://127.0.0.1:5000/existing\_user' }}" method="POST" role="form">

<button type="submit" class="btn btn-success">Existing user Login</button>

</form>

<br>

<br>

</div>

</body>

</html>

New\_user.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>New User</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<h1> Thank you for choosing our hospital!</h1>

<br>

<form action="{{ 'http://127.0.0.1:5000/insert\_patient\_data' }}" method="POST" role="form">

<div class="form-group">

<label for="id\_of\_user">Enter the Health ID you want:</label>

<input type="text" class="form-control" id="id\_of\_user" name="id\_of\_user" placeholder="a456">

<br>

<br>

<label for="name\_of\_user">Enter your name:</label>

<input type="text" class="form-control" id="name\_of\_user" name="name\_of\_user" placeholder="Elon Musk" required autocomplete="off">

<br>

<br>

<label for="email\_of\_user">Enter your email:</label>

<input type="text" class="form-control" id="email\_of\_user" name="email\_of\_user" placeholder="elon.musk@tesla.com">

<br>

<br>

<label for="ssn\_of\_user">Your SSN:</label>

<input type="text" class="form-control" id="ssn\_of\_user" name="ssn\_of\_user" placeholder="123456789">

<br>

<br>

<label for="psw\_of\_user"> Enter your Password:</label>

<input type="password" class="form-control" id="psw\_of\_user" name="psw\_of\_user" placeholder="Enter a password.">

</div>

<button type="submit" class="btn btn-success">Sign Up</button>

</form>

<br>

</div>

<br>

</body>

</html>

New\_user\_success.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Sucess Page</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<br>

<br>

<h1>{{ msg }}</h1>

<br>

<h3>Go to the main page to login as existing user.</h3>

<form action="{{ 'http://127.0.0.1:5000/success' }}" method="POST" role="form">

<div class="form-group">

</div>

<br>

<button type="submit" class="btn btn-success">Go to the Main Page.</button>

</form>

<br>

</div>

<br>

</body>

</html>

Patient.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Patient Page</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<h3>Hello {{ result[1] }}</h3>

<br>

<br>

<p> Please select any of the options below </p>

<br>

<br>

<form action="{{ 'http://127.0.0.1:5000/update\_patient\_info' }}" method="POST" role="form">

<button type="submit" class="btn btn-success">Update my information</button>

</form>

<br>

<br>

<form action="http://127.0.0.1:5000/book\_appointment" method="POST" role="form">

<button type="submit" class="btn btn-success">Book Appointment</button>

</form>

<br>

<br>

<form action="http://127.0.0.1:5000/cancel\_appointment" method="POST" role="form">

<button type="submit" class="btn btn-success">Cancel Appointment</button>

</form>

<br>

<br>

<form action="http://127.0.0.1:5000/view\_appointment" method="POST" role="form">

<button type="submit" class="btn btn-success">View my Appointments</button>

</form>

<br>

<br>

<form action="http://127.0.0.1:5000/scan\_report" method="POST" role="form">

<button type="submit" class="btn btn-success">Get Scan report</button>

</form>

<br>

</div>

<br>

</body>

</html>

Scan\_report\_success.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Sucess/Failure Page</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<br>

<br>

<h1>{{ msg }}</h1>

<form action="{{ 'http://127.0.0.1:5000/success' }}" method="POST" role="form">

<div class="form-group">

</div>

<br>

<button type="submit" class="btn btn-success">Go to the Main Page.</button>

</form>

<br>

</div>

<br>

</body>

</html>

Update\_patient\_email.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Update Your Email</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<br>

<form action="{{ 'http://127.0.0.1:5000/update\_patient\_email\_success' }}" method="POST" role="form">

<div class="form-group">

<br>

<label for="email\_of\_user">Your new Email:</label>

<input type="text" class="form-control" id="email\_of\_user" name="email\_of\_user" placeholder="elon.musk@tesla.com" required autocomplete="off">

<br>

</div>

<button type="submit" class="btn btn-success">Update</button>

</form>

<br>

</div>

<br>

</body>

</html>

Update\_patient\_info\_page.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Update Patient info Page</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<br>

<br>

<p> Please select any of the options below </p>

<br>

<br>

<form action="{{ 'http://127.0.0.1:5000/update\_patient\_name' }}" method="POST" role="form">

<button type="submit" class="btn btn-success">Change my name</button>

</form>

<br>

<br>

<form action="http://127.0.0.1:5000/update\_patient\_email" method="POST" role="form">

<button type="submit" class="btn btn-success">Update Email</button>

</form>

<br>

<br>

<form action="http://127.0.0.1:5000/update\_patient\_ssn" method="POST" role="form">

<button type="submit" class="btn btn-success">Update SSN</button>

</form>

<br>

</div>

<br>

</body>

</html>

Update\_patient\_name.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Update Your Email</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<br>

<form action="{{ 'http://127.0.0.1:5000/update\_patient\_name\_success' }}" method="POST" role="form">

<div class="form-group">

<br>

<label for="name\_of\_user">Enter your name:</label>

<input type="text" class="form-control" id="name\_of\_user" name="name\_of\_user" placeholder="Elon Musk" required autocomplete="off">

<br>

</div>

<button type="submit" class="btn btn-success">Update</button>

</form>

<br>

</div>

<br>

</body>

</html>

Update\_patient\_ssn.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>Update Your Email</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<br>

<form action="{{ 'http://127.0.0.1:5000/update\_patient\_ssn\_success' }}" method="POST" role="form">

<div class="form-group">

<br>

<label for="ssn\_of\_user">Your new SSN:</label>

<input type="text" class="form-control" id="ssn\_of\_user" name="ssn\_of\_user" placeholder="Elon Musk" required autocomplete="off">

<br>

</div>

<button type="submit" class="btn btn-success">Update</button>

</form>

<br>

</div>

<br>

</body>

</html>

uploadFile.html

<!doctype html>

<title>Upload new File</title>

<h1>Upload scan report</h1>

<form method=post enctype=multipart/form-data>

<p><input type=file name=file>

<input type=submit value=Upload>

</p>

<label for="patient\_id\_file\_upload"><b>Patient health ID: </b></label>

<input type="text" name="patient\_id\_file\_upload" required>

</form>

view\_appointment.html

<!DOCTYPE html>

<html>

<head>

<title>Your appointments</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div class="container">

<h2>This is your appointment.</h2>

<div class="table-responsive">

<table class="table">

<thead>

<tr>

<th>Name</th>

<th>Doctor</th>

<th>Appointment</th>

</tr>

</thead>

<tbody>

<tr>

{% for row in result %}

<td>{{ row[1] }}</td>

<td>{{ row[7] }}</td>

<td>{{ row[4] }}</td>

{% endfor %}

</tr>

</tbody>

</table>

</div>

<br>

</div>

</body>

</html>

Winsoc.c

#define WIN32\_LEAN\_AND\_MEAN

#include <windows.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <stdlib.h>

#include <stdio.h>

void WSAAPI freeaddrinfo( struct addrinfo\* );

int WSAAPI getaddrinfo( const char\*, const char\*, const struct addrinfo\*,

struct addrinfo\*\* );

int WSAAPI getnameinfo( const struct sockaddr\*, socklen\_t, char\*, DWORD,

char\*, DWORD, int );

// Need to link with Ws2\_32.lib, Mswsock.lib, and Advapi32.lib

#pragma comment (lib, "Ws2\_32.lib")

#pragma comment (lib, "Mswsock.lib")

#pragma comment (lib, "AdvApi32.lib")

#define LINELENGTH 65535

#define DEFAULT\_PORT "80"

int \_\_cdecl main(int argc, char \*\*argv)

{

WSADATA wsaData;

SOCKET ConnectSocket = INVALID\_SOCKET;

struct addrinfo \*result = NULL,

\*ptr = NULL,

hints;

char \*sendbuf = "/";

// char recvbuf[DEFAULT\_BUFLEN];

int iResult;

//int recvbuflen = DEFAULT\_BUFLEN;

char\* request;

char\* request\_post;

char\* request\_post1;

char \*filepath;

char buf[LINELENGTH+1];

char buf\_read[LINELENGTH+1];

char buf\_read1[LINELENGTH+1];

char\* server = "localhost";

int w,reading;

char\* header = "Host: 192.168.1.5\r\nUser-Agent: Chrome/5.0 (X11; Linux x86\_64; rv:52.0) Gecko/20100101 Firefox/52.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\n"

"Connection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\nContent-Type: application/x-www-form-urlencoded\r\n"

"Content-Length: 23\r\n\r\n";

char\* header1 = "Host: 192.168.1.5\r\nUser-Agent: Chrome/5.0 (X11; Linux x86\_64; rv:52.0) Gecko/20100101 Firefox/52.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\n"

"Connection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\n\r\n";

char\* header2 = "Host: 127.0.0.1\r\nUser-Agent: Chrome/5.0 (X11; Linux x86\_64; rv:52.0) Gecko/20100101 Firefox/52.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\n"

"Connection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\nContent-Type: application/x-www-form-urlencoded\r\n"

"\r\n";

//Memory allocation for pointers

filepath = malloc(256);

request = malloc(65536);

request\_post = malloc(65536);

request\_post1 = malloc(65536);

if(argc == 3){

filepath = argv[1];

server = argv[2];

}

else if (argc == 2){

printf("File not entered. Connecting to server without any file request\n");

filepath = "nofile";

server = argv[1];

}

else

{

printf("please enter file and server details \n");

exit(1);

}

// Initialize Winsock

iResult = WSAStartup(MAKEWORD(2,2), &wsaData);

if (iResult != 0) {

printf("WSAStartup failed with error: %d\n", iResult);

return 1;

}

ZeroMemory( &hints, sizeof(hints) );

hints.ai\_family = AF\_UNSPEC;

hints.ai\_socktype = SOCK\_STREAM;

hints.ai\_protocol = IPPROTO\_TCP;

// Resolve the server address and port

iResult = getaddrinfo(argv[1], DEFAULT\_PORT, &hints, &result);

if ( iResult != 0 ) {

printf("getaddrinfo failed with error: %d\n", iResult);

WSACleanup();

return 1;

}

// Attempt to connect to an address until one succeeds

for(ptr=result; ptr != NULL ;ptr=ptr->ai\_next) {

// Create a SOCKET for connecting to server

ConnectSocket = socket(ptr->ai\_family, ptr->ai\_socktype,

ptr->ai\_protocol);

if (ConnectSocket == INVALID\_SOCKET) {

printf("socket failed with error: %ld\n", WSAGetLastError());

WSACleanup();

return 1;

}

// Connect to server.

iResult = connect( ConnectSocket, ptr->ai\_addr, (int)ptr->ai\_addrlen);

if (iResult == SOCKET\_ERROR) {

closesocket(ConnectSocket);

ConnectSocket = INVALID\_SOCKET;

printf("Socket error \n");

continue;

}

break;

}

// freeaddrinfo(result);

if (ConnectSocket == INVALID\_SOCKET) {

printf("Unable to connect to server!\n");

WSACleanup();

return 1;

}

printf("connection sucessful \n");

//Creating an http request

FILE \*fptr;

fptr = fopen("loginfile", "w");

memset(request\_post,0, 65536);

strcat(request\_post,"GET ");

strcat(request\_post, "/ HTTP/1.1\r\n");

strcat(request\_post, header1);

printf(request\_post);

w = send(ConnectSocket,request\_post,(int)strlen(request\_post),0);

if (w == SOCKET\_ERROR) {

printf("send failed with error: %d\n", WSAGetLastError());

closesocket(ConnectSocket);

WSACleanup();

return 1;

}

printf("Bytes Sent: %ld\n", w);

/\* while((reading = recv(ConnectSocket, buf, LINELENGTH+1,0))>0)

{

printf("read bytes %d\n",reading);

fwrite(buf,1,reading,fptr);

memset(buf,0, 65536);

}\*/

reading = recv(ConnectSocket, buf, LINELENGTH+1,0);

fwrite(buf,1,reading,fptr);

memset(buf,0, 65536);

printf(buf);

printf("\n");

fclose(fptr);

memset(buf, 0, sizeof(buf));

reading = 0;

FILE \*fptr1;

fptr1 = fopen("existing user", "w");

memset(request\_post1,0, 65536);

strcat(request\_post1,"POST ");

strcat(request\_post1, "/existing\_user HTTP/1.1\r\n");

strcat(request\_post1, header2);

//printf(request\_post1);

//printf("len = %d\n", strlen(request\_post1));

w = send(ConnectSocket,request\_post1,(int)strlen(request\_post1),0);

/\*while((reading = recv(ConnectSocket, buf, LINELENGTH+1,0))>0)

{

printf("REad size = %d \n", reading);

fwrite(buf,1,reading,fptr1);

memset(buf,0,LINELENGTH+1);

}\*/

reading = recv(ConnectSocket, buf, LINELENGTH+1,0);

fwrite(buf,1,reading,fptr1);

memset(buf,0,LINELENGTH+1);

fclose(fptr1);

memset(buf,0, LINELENGTH+1);

reading = 0;

FILE \*fptr2;

fptr2 = fopen("patient.html", "w");

//Writing to server

memset(request,0, 65536);

strcat(request,"POST ");

strcat(request, "/result HTTP/1.1\r\n");

strcat(request, header);

strcat(request,"uname=p0123&psw=prakash");

printf("\n");

printf(request);

int counter = 0;

w = send(ConnectSocket,request,(int)strlen(request),0);

if(w<0){

printf("write not happening \n");

}

else{

printf("%d bytes written by client\n", w);

}

//Reading response from the server

/\* while((reading = recv(ConnectSocket, buf, LINELENGTH+1,0))>0)

{

printf(buf);

fwrite(buf,1,reading,fptr2);

memset(buf,0,LINELENGTH+1);

}\*/

reading = recv(ConnectSocket, buf, LINELENGTH+1,0);

printf(buf);

printf("bytes read %d \n", reading );

fwrite(buf,1,reading,fptr2);

fclose(fptr2);

// Send an initial buffer

/\* iResult = send( ConnectSocket, sendbuf, (int)strlen(sendbuf), 0 );

if (iResult == SOCKET\_ERROR) {

printf("send failed with error: %d\n", WSAGetLastError());

closesocket(ConnectSocket);

WSACleanup();

return 1;

}\*/

// shutdown the connection since no more data will be sent

iResult = shutdown(ConnectSocket, SD\_SEND);

if (iResult == SOCKET\_ERROR) {

printf("shutdown failed with error: %d\n", WSAGetLastError());

closesocket(ConnectSocket);

WSACleanup();

return 1;

}

// Receive until the peer closes the connection

/\* do {

iResult = recv(ConnectSocket, recvbuf, recvbuflen, 0);

if ( iResult > 0 )

{

printf("Bytes received: %d\n", iResult);

printf(recvbuf);

}

else if ( iResult == 0 )

printf("Connection closed\n");

else

printf("recv failed with error: %d\n", WSAGetLastError());

} while( iResult > 0 );\*/

// cleanup

closesocket(ConnectSocket);

WSACleanup();

return 0;

}

Webclient.c (BSD sockets)

#include<stdio.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netdb.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<stdarg.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

#include<errno.h>

#include<ctype.h>

// defining INADDR\_NONE if its not already defined

#ifndef INADDR\_NONE

#define INADDR\_NONE 0xffffffff

#endif

extern int errno;

/\*

\* This function is defined to connect the client to the server

\* Arguments

\* server: server to which the host is to be connected

\* service: the name of service client needs

\* transport: the transport protocol through which client is sending requests

\*/

int connectTCP (char \*server, char \*transport, char \*portnumber);

/\*

\* Used to print error message and exits. Funtion can accept variable number of arguments

\* Argument:

\* format: string type variable describing user given error description

\*

\*/

int errexit(const char \*format, ...);

/\*

\* Checks whether the string passed as parameter is a number or not

\* Also validates whether its positive or negative

\* if negative the function exits as square root of negative number cannot be found

\*/

#define LINELENGTH 65535

int main(int argc, char \*argv[])

{

char\* server = "localhost";

char\* transport = "tcp";

char \*portnumber = "80";

int sock,n,w;

char\* request;

char\* request\_post;

char\* request\_post1;

char \*filepath;

char buf[LINELENGTH+1];

char buf\_read[LINELENGTH+1];

char buf\_read1[LINELENGTH+1];

char\*file;

int reading;

int read\_total = 0;

int size = LINELENGTH+1;

//char\* header = "Host:Webclient\r\nContent-Language:en\r\nContent-Type:application/x-www-form-url-encoded\r\nContent-Length:15\r\n\r\n";

char\* header = "Host: 127.0.0.1\r\nUser-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:52.0) Gecko/20100101 Firefox/52.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\n"

"Connection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\nContent-Type: application/x-www-form-urlencoded\r\n"

"Content-Length: 23\r\n\r\n";

char\* header1 = "Host: 127.0.0.1\r\nUser-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:52.0) Gecko/20100101 Firefox/52.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\n"

"Connection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\n\r\n";

char\* header2 = "Host: 127.0.0.1\r\nUser-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:52.0) Gecko/20100101 Firefox/52.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\n"

"Connection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\nContent-Type: application/x-www-form-urlencoded\r\n"

"\r\n";

//Memory allocation for pointers

filepath = malloc(256);

request = malloc(65536);

request\_post = malloc(65536);

request\_post1 = malloc(65536);

//Validation to check if the inputs are properly entered

if(argc == 3){

filepath = argv[1];

server = argv[2];

}

else if (argc == 2){

printf("File not entered. Connecting to server without any file request\n");

filepath = "nofile";

server = argv[1];

}

else

{

printf("please enter file and server details \n");

exit(1);

}

//Connecting to server

sock = connectTCP(server, transport, portnumber);

/\*strcat(request\_post, "GET / HTTP/1.1\r\n");

strcat(request\_post, header1);

strcat(request\_post, "\r\n");\*/

FILE \*fptr;

fptr = fopen("loginfile", "w");

strcat(request\_post,"GET ");

strcat(request\_post, "/ HTTP/1.1\r\n");

strcat(request\_post, header1);

printf(request\_post);

//printf("len = %d", strlen(request\_post));

w = write(sock,request\_post,strlen(request\_post));

printf("write done and reading happening\n");

reading = read(sock, buf, LINELENGTH+1);

printf("read");

fwrite(buf,1,reading,fptr);

fclose(fptr);

memset(buf, 0, sizeof(buf));

reading = 0;

FILE \*fptr1;

fptr1 = fopen("existing user", "w");

strcat(request\_post1,"POST ");

strcat(request\_post1, "/existing\_user HTTP/1.1\r\n");

strcat(request\_post1, header2);

printf(request\_post1);

printf("len = %d\n", strlen(request\_post1));

w = write(sock,request\_post1,strlen(request\_post1));

reading = read(sock, buf, LINELENGTH+1);

fwrite(buf,1,reading,fptr1);

fclose(fptr1);

memset(buf,0, LINELENGTH+1);

reading = 0;

FILE \*fptr2;

fptr2 = fopen("patient.html", "w");

//Writing to server

strcat(request,"POST ");

strcat(request, "/result HTTP/1.1\r\n");

strcat(request, header);

strcat(request,"uname=p0123&psw=prakash");

printf("\n");

printf(request);

int counter = 0;

w = write(sock,request,strlen(request));

if(w<0){

errexit("write not happening %s \n", strerror(errno));

}

else{

printf("%d bytes written by client\n", w);

}

//Reading response from the server

reading = read(sock, buf, LINELENGTH+1);

printf(buf);

printf("bytes read %d \n", reading );

fwrite(buf,1,reading,fptr2);

fclose(fptr);

close(sock);

return 0;

}

int connectTCP (char \*server, char \*transport, char \*portnumber){

int counter\_num = 0;

int counter\_ip = 0;

int counter\_hostname = 0;

char \*ip;

char \*hostname;

int sock,type;

int optval;

socklen\_t optlen = sizeof(optval);

struct sockaddr\_in sin;

struct servent \*pse;

struct hostent \*phe;

struct protoent \*ppe;

//validating if portnumber is numeric

for(int i = 0; i< strlen(portnumber); i++){

if(!(isdigit(portnumber[i]))){

errexit("error portnumber is not all numeric \n");

}

}

fprintf(stdout, "port number entered is all numeric \n");

//validating if server name passed is all numeric

/\* for(int i = 0; i<strlen(server); i++){

if(isdigit(server[i])){

counter\_num++;

}

}\*/

/\* if(counter\_num == strlen(server)){

errexit("host name or host ip cannot be all numeric \n");

}\*/

///validating if the server input is ip

/\*for(int j = 0; j<strlen(server); j++){

if((isdigit(server[j])) || (server[j] == '.')){

printf("ip is %s \n", server[j]);

counter\_ip++;

}

else{

break;

}

}\*/

//validating if the server input is hostname

/\* for(int i = 0; i< strlen(server); i++){

if((isalnum(server[i])) || ispunct(server[i])){

counter\_hostname++;

}

else{

break;

}

}\*/

//assigning values

/\* if(counter\_ip == strlen(server)){

ip = server;

}

else if(counter\_hostname == strlen(server)){

hostname = server;

}

else{

errexit("value input for hostname/ip is not in the correct format \n");

}\*/

//setting all values to 0 in sockaddr\_in

memset(&sin, 0 , sizeof(sin));

//setting family name and portnumber

sin.sin\_family = AF\_INET;

//converting portnumber to integer

uint16\_t port =(uint16\_t) atoi(portnumber);

sin.sin\_port = htons(port);

//setting the type of packet

if(transport == "tcp"){

type = SOCK\_STREAM;

}

else if(transport == "udp"){

type = SOCK\_DGRAM;

}

else{

errexit("transport type couldnot be matched \n");

}

//mapping hostname or ip addresses

ip = server;

if(inet\_addr(ip) != INADDR\_NONE){

sin.sin\_addr.s\_addr = inet\_addr(ip);

fprintf(stdout, "ip address successfully mapped \n");

}

else{

errexit("ip address given is not valid %s \n", strerror(errno));

}

/\*else if(hostname){

phe = gethostbyname(hostname);

if(phe){

memcpy(&sin.sin\_addr.s\_addr, phe->h\_addr, phe->h\_length);

fprintf(stdout, "hostname successfully mapped to ip address \n");

}

else if((sin.sin\_addr.s\_addr = inet\_addr(hostname)) == INADDR\_NONE){

errexit("host name is not correct %s \n", strerror(errno));

}

}\*/

//Validating protocol number

if((ppe = getprotobyname(transport)) == 0){

errexit("Cant get the protocol entry for protocol %s \n" , transport);

}

//Creating socket

sock = socket(PF\_INET, type, ppe ->p\_proto);

if(sock < 0){

errexit("Socket not created properly %s \n", strerror(errno));

}

//Setting the keep alive option for the socket

if(getsockopt(sock,SOL\_SOCKET, SO\_KEEPALIVE, &optval, &optlen) < 0)

{

printf("error in getting the keep alive option \n");

exit(1);

}

if(optval == 0)

{

printf("Keep alive option not set for socket \n");

optval = 1;

if(setsockopt(sock, SOL\_SOCKET, SO\_KEEPALIVE, &optval, optlen) < 0)

{

printf("error in setting the keep alive option \n");

exit(1);

}

else

{

printf("success setting keep alive option \n ");

}

}

//Connecting to server

if(connect(sock, (struct sockaddr \*)&sin, sizeof(sin)) <0){

errexit("connection to server failed %s \n",strerror(errno));

}

else{

fprintf(stdout,"connected to server sucesssfully \n");

}

return sock;

}

int errexit(const char \*format, ...){

va\_list args;

va\_start(args, format);

vfprintf(stderr, format, args);

va\_end(args);

exit(1);

}

1. **PRESENTATION**