NEW HORIZON COLLEGE OF ENGINEERING

New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

A

MINI PROJECT REPORT

ON

"MEDITRACKER"

Submitted in the partial fulfillment of the requirements in the VI semester of

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

BY

AKHILA S - [1NH17IS008]

COURSE NAME: MINI PROJECT
COURSE CODE: ISE66

Under the guidance of

Ms Shobha Shanmugham
Sr. Assistant Professor
Dept. of ISE, NHCE

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

NEW HORIZON COLLEGE OF ENGINEERING
(Autonomous College Permanently Affiliated to VTU, Approved by AICTE, Accredited by NBA & NAAC with 'A' Grade)
New Horizon Knowledge Park, Ring Road, Bellandur Post, Near Marathalli,
Bangalore-560103, INDIA

NEW HORIZON
COLLEGE OF ENGINEERING
New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

I hereby certify that, the report entitled **"Meditracker"** as a part of Mini Project Component in partial fulfillment of the requirements during 6th semester Bachelor of Engineering in Information Science and Engineering during the year 2019-20(Jan 2020-May 2020) is an authentic record of my own work carried out by **Akhila S (1NH17IS008)**, a bonafied student of NEW HORIZON COLLEGE OF ENGINEERING.

Name & Signature of Student

(Ms. Akhila S)

Name & Signature of Guide

(Ms. Shobha Shanmugam)

Name & signature of HOD

(Dr. R J Anandhi)

PLAGARISM CERTIFICATE

MEDITRACKER

ORIGINALITY REPORT

2%	%	2%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	W. J. Buchanan. "Chapter 95 ASP", Springer Science and Business Media LLC, 2004 Publication	2%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

ABSTRACT

Medi tracker is an application which is used by both doctors and patients. It's very tough for some people to remember what medicine has to be taken at what time. So this makes our work easier. The main aim of this application is it acts like a health assistant. Not only does it assesses your health but also acts as a great tool when you visit the doctor. The features are reminders of the medications, monitoring prescriptions given to the patients, fixing doctor's appointment, keeping all your health records in one place including the doctor's contact, details of your health insurance and also give a report of all your health activities. One of the main flaws of the existing system is that the records are maintained manually and we tend to forget about our medicines as we are busy. Hence the user or the patient can track his medical records whenever required. A web application is developed for this project .The backend will be a database (MYSQL), middle tier using Python and the front end will be developed using HTML, CSS, and JavaScript.

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped me in carrying out this mini project. I would like to take this opportunity to thank them all.

I thank the management, **Dr. Mohan Manghnani,** Chairman, New Horizon Educational Institutions for providing necessary infrastructure and creating conducive environment for effective learning.

I also record here the constant encouragement, support and facilities extended to us by **Dr. Manjunatha,** Principal, New Horizon College of Engineering, Bengaluru.

I extent sincere gratitude for constant encouragement and facilities provided to us by **Dr. R.J Anandhi**, Professor and Head of the Department, Department of Information Science and Engineering, New Horizon College of Engineering, Bengaluru.

I sincerely acknowledge the encouragement, timely help and guidance to me by **Ms. Shobha Shanmugham,** Sr. Assistant Professor, Department of Information Science and Engineering, New Horizon College of Engineering, Bengaluru, to complete the mini project within stipulated time successfully.

Finally, a note of thanks to the teaching and non-teaching staff of Information Science and Engineering Department for their cooperation extended to us and our friends, who helped me directly or indirectly in the successful completion of this mini project.

Akhila S

(1NH17IS008)

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Meditracker is an e service provided to both doctors as well as patients with easy to use customizable options. The application will basically lessen the manual work and improve the quality of maintaining records and other information related to doctors or patients. It reduces time frame in adding any information related to hospital and thereby reduce the complexity too. The application allows patients to book appointment with doctors based on his availability. The application consists of patient module where they can view their records, book an appointment. Doctor module where he can view the patient's medical history and note down their vital as well. And prescribe them with medicines, modify his appointment availability and view his appointments. Admin module where they can add the records of new patient, delete their records, update them.

1.1 Motivation of Project

The purpose of creating a Meditracker is to add, edit, delete and update the patient record. It is also used to book appointments with doctors, view the doctor's availability and the doctor can also change his availability status. This reduces a lot of manual work. If we look at traditional methods, we understand that manual approach is needed where the records are handwritten and there are chances that the data entered might be wrong. The computerized accounting of data offers unlimited storage of memory and the information retrieval is also easy. It is one of the best improvements of the traditional book keeping method.

The data is stored in the form of tables which has proven easy to comprehend and effective in the long run. The hospital also must maintain a proper record of all the patients in a database so that it is easily accessible in the future. Apart from that the website created is user friendly.

1.2 Methodology

FRONTEND:-

Frontend is a term used to build webpages and user interfaces for web application. It is the part of the website where the client can interact. All the navigation links, button, font colors, form is a combination of HTML, CSS and JavaScript. Apart from these front end technologies, there are frameworks like Bootstrap, AngularJS, Foundation and so on which can be used to create a website. In this project the website is being used by 3 different types of users or clients- Admin, Doctor and Patient.

MIDDLE TIER:-

Flask is a micro web framework written in Python. It was created by Armin Ronacher of Pocoo. The key aspects of flask development are: how to integrate flask applications with front end, jinja templating, Sessions, API Calls, databases, managing HTTP requests.

BACKEND:-

We use MYSQL database as the backend in this project. MYSQL is open source software where we create the database. The data is stored in the form of tables. The data from these tables can be accessed using mysql connector.

1.3 Problem Statement

To design and implement an application which allows the user to track his/her medical records and fix appointments with the doctor based on his availability. Also supports Doctor to reschedule and manage his appointments.

Chapter 2

SYSTEM REQUIREMENT & LANGUAGE USED

Purpose: To design and implement an application which allows the user to track his/her medical records and fix appointments with the doctor based on his availability. Also supports Doctor to reschedule and manage his appointments.

2.1 Hardware System Configuration:

Processor	- Intel core i5
Speed	- 1.8 GHz
RAM	- 256 MB (min)
Hard Disk	- 10 GB

2.2 Software System Configuration:

Operating System	-	Windows 8.1
Programming Language	-	Python
Compiler	-	PyCharm

2.2 About the language used

PyCharm is an open source distribution of Python and R programming which is used for graphical debugger, an integrated unit tester, supports web development with Django, machine learning, data science, and code analysis. In this project we make use of PyCharm IDE to write our code in Python. Python is a high level programming language which makes use of object oriented programming concepts.

Features of Python language are:

1. It is user friendly and easy to understand.
2. It is readable.
3. It is platform independent; it can work on any operating system.
4. It is open source.
5. It is an interpreted language where debugging happens line by line.

Chapter 3

METHODOLOGY

3.1 Architecture

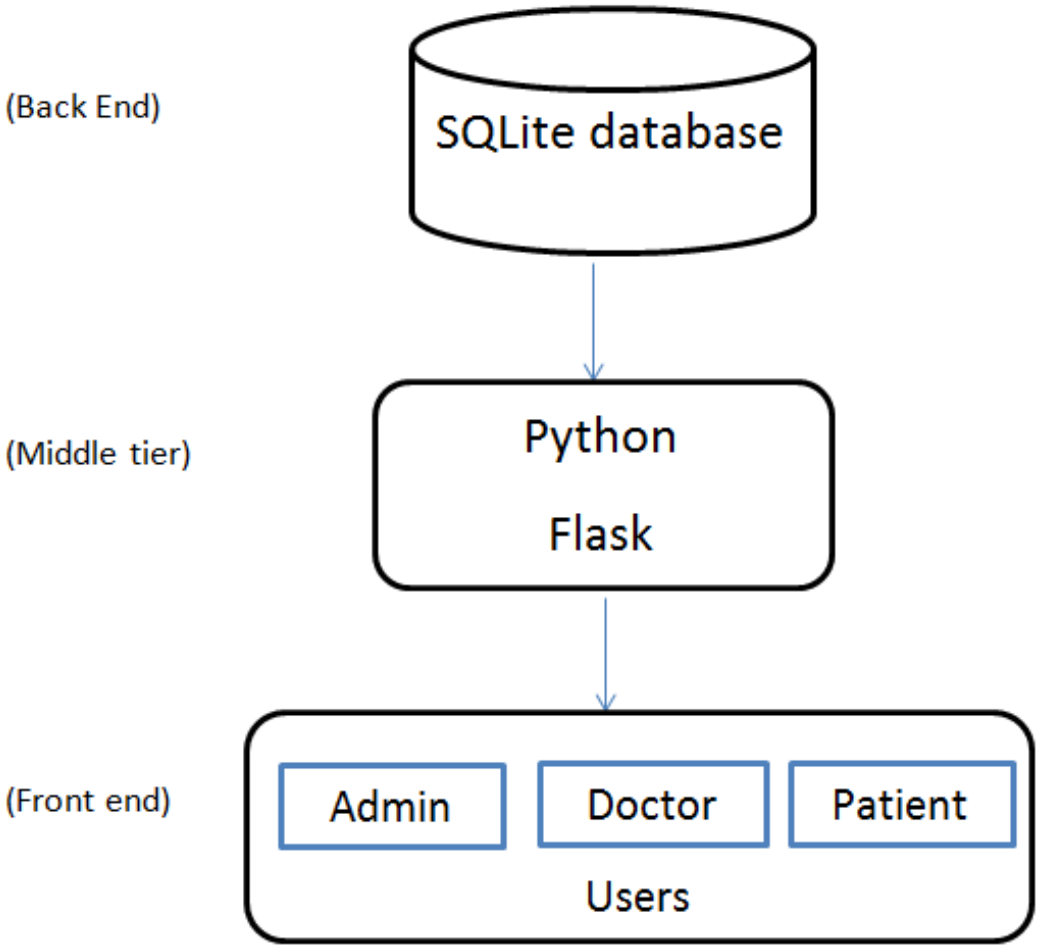


Fig 3.1 Architecture

3.2 Algorithm

Step 1: Start

Step 2: Register the details of the user on the website created and store it in a database.

Here the user can be of three types:

a) Admin
b) Doctor
c) Patient

Step 3: The user must login using their registered credentials in order to use the application.

Step 4: When the user logins, the email id and password is verified with the data present in the database. If the details don't match, an appropriate error message is shown.

Step 5: Once the user has logged in, the website will redirect to its respective page based on the user type that is admin will be redirected to admin page and so on.

Step 6: The user can perform these functions after log in:

ADMIN:-

a) Add Patient Record
b) View Patient Record
c) Edit Patient Record
d) View Doctor Schedule

DOCTOR:-

a) View Patient Record
b) Edit Patient Record
c) Add Schedule
d) View Appointments made

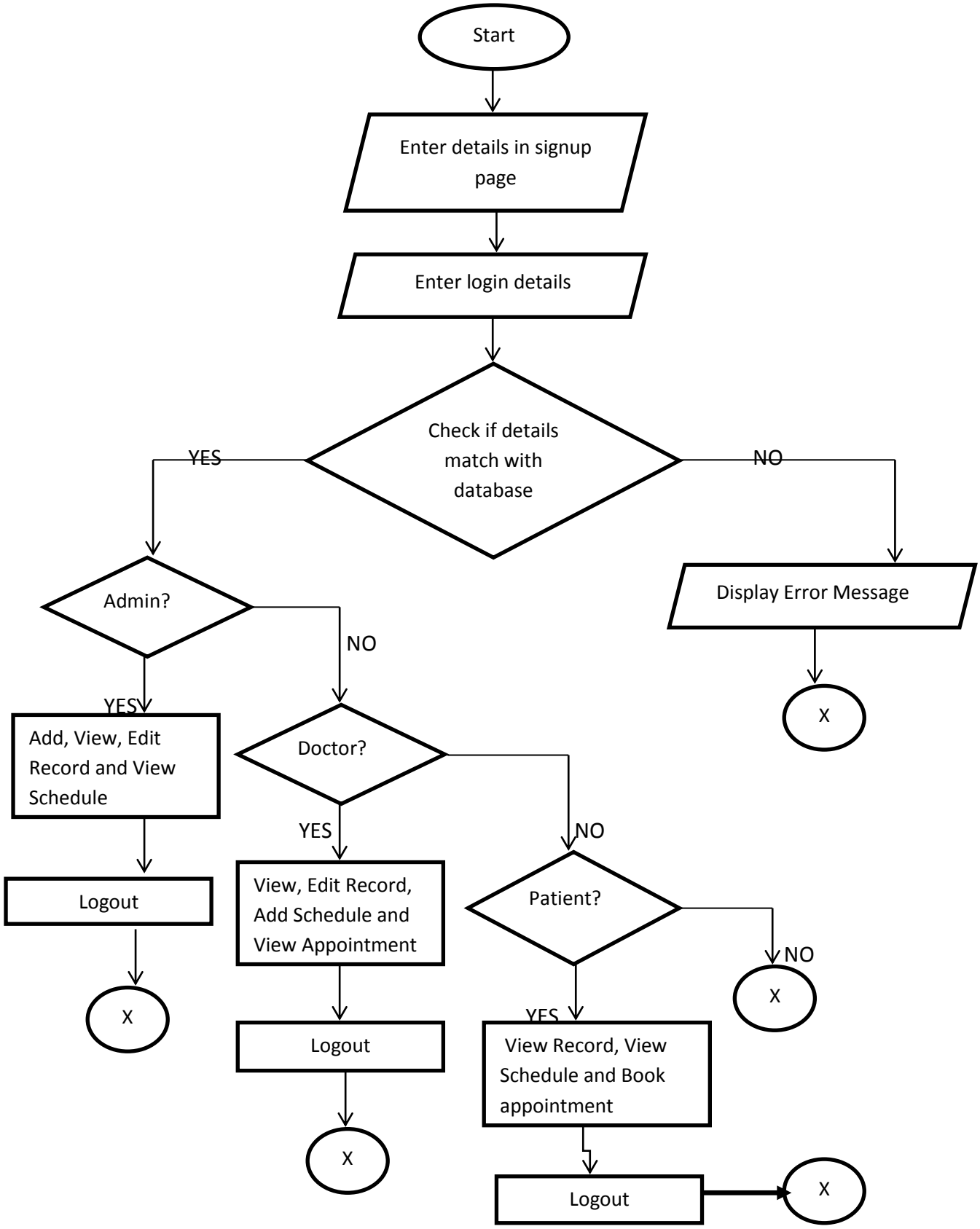
PATIENT:-

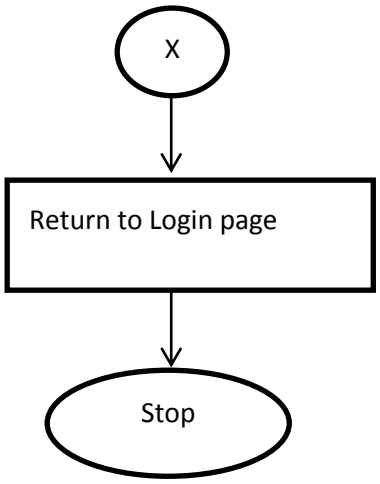
a) View Patient Record
b) View Doctor's Schedule
c) Book Appointments

Step 7: Once the user finishes their task, they can log out.

Step 8: Stop

3.3 Flowchart




X
Return to Login page
Stop

3.4 Code and Implementation

```
from flask import Flask, render_template, request, redirect,session,flash,url_for,g

from datetime import  timedelta

app=Flask(__name__)

import os

import mysql.connector

app.secret_key = os.urandom(24)

app.permanent_session_lifetime = timedelta(minutes=5)

mydb=mysql.connector.connect(

   host='localhost',

   user='root',

   passwd='',

   database = 'meditracker'

)

@app.before_request

def before_request():

   g.user = None

   if 'user' in session:

      g.user=session['user']

   print(g.user)

@app.route("/")
```

```
def home():
   return render_template('index.html')
@app.route("/about")
def about():
   return render_template('about.html')
@app.route("/contact", methods=['GET', 'POST'])
def contact():
   if(request.method=='POST'):
      name=request.form.get('name')
      type = request.form.get('type')
      phone=request.form.get('phone')
      email=request.form.get('email')
      password=request.form.get('password')
      query="insert into  signup(name,type,phone,email,password) VALUES
(%s,%s,%s,%s,%s)"
      val = (name,type,phone,email,password)
      mycursor = mydb.cursor()
      mycursor.execute(query, val)
      mydb.commit()
   return render_template('contact.html')
@app.route("/login", methods=['GET','POST'])
def login():
```

```
    if(request.method=='POST'):
        session.pop('user', None)
        email=request.form.get('email')
        password=request.form.get('password')
        try:
            mycursor = mydb.cursor()
            mycursor.execute('SELECT * FROM signup WHERE email=%s and password=%s',
(email,password))
            records = mycursor.fetchall()
            session['user'] = email
            # session.permanent = True
            if "user" in session :
                user = session['user']
            for row in records:
                type=str(row[1])
            if(type=='admin'):
                return redirect(url_for('admin'))
            if(type=='patient'):
                return redirect(url_for('patient'))
            if(type=='doctor'):
                return redirect(url_for('doctor'))
        except:
```

```
            print("Wrong email or password")
            flash("Wrong Details")
            return render_template("login.html")
    try:
        return render_template('login.html',user=user)
    except:
        return render_template('login.html')
@app.route('/admin')
def admin():
    if(g.user):
        return render_template('admin1.html',user=session['user'])
    return render_template('login.html')
@app.route('/addrec')
def addrec():
    if "user" in session :
        user = session['user']
    return render_template('add.html',user=user)
@app.route('/addrecpost', methods=['POST','GET'])
def addrecpost():
    if 'user' in session:
        user = session['user']
```

```
    if (request.method == 'POST') :
        name=request.form.get('name')
        email=request.form.get('email')
        dob=request.form.get('dob')
        gender=request.form.get('gender')
        contact=request.form.get('contact')
        medical=request.form.get('medical')
        print("1")
        query = "insert into  addreco(name,email,dob,gender,contact,medical) VALUES
(%s,%s,%s,%s,%s,%s)"
        val = (name,email, dob, gender, contact, medical)
        mycursor = mydb.cursor()
        mycursor.execute(query, val)
        print("2")
        mydb.commit()
    return render_template('add.html')
@app.route("/viewrec",methods=['POST','GET'])
def view():
    if 'user' in session :
        user = session['user']
    if (request.method == 'POST') :
```

```
        try:
            email=request.form.get("email")
            mycursor=mydb.cursor()
            mycursor.execute('SELECT * FROM addreco WHERE email=%s',(email,))
            myresult=mycursor.fetchall()
            for x in myresult:
                print(x)
            mydb.commit()
        except:
            render_template('view.html',user=user)
    try:
        return render_template('view.html',data=myresult,user=user)
    except:
        return render_template('view.html',  user=user)
@app.route("/editrec",methods=['POST','GET'])
def edit():
    if 'user' in session :
        user = session['user']
    if (request.method == 'POST') :


        try:
```

```
            email=request.form.get("email")
            mycursor=mydb.cursor()
            mycursor.execute('SELECT * FROM addreco WHERE email=%s',(email,))
            myresult=mycursor.fetchall()
            for x in myresult:
                print(x)
            mydb.commit()
        except:
            render_template('edit.html',user=user)
    try:
        return render_template('edit.html',data=myresult,user=user)
    except:
        return render_template('edit.html',  user=user)
@app.route('/update')
def upd():
    if(g.user):
        return render_template('update.html',user=session['user'])
    return render_template('update.html')
@app.route('/updaterec', methods=['POST', 'GET'])
def update():
    if request.method=='POST':
```

```
        name=request.form.get('name')
        email=request.form.get('email')
        dob=request.form.get('dob')
        gender=request.form.get('gender')
        contact=request.form.get('contact')
        medical=request.form.get('medical')
        cur=mydb.cursor()
        cur.execute(""" UPDATE addreco SET name=%s, email=%s, dob=%s, gender=%s,
contact=%s, medical=%s """,(name,email,dob,gender,contact,medical))
        flash("DATA UPDATED SUCCESSFULLY")
        mydb.commit()
    return render_template('edit.html')
@app.route("/viewsch",methods=['POST','GET'])
def viewsch():
    if 'user' in session :
        user = session['user']
    if (request.method == 'POST') :

        try:
            docname=request.form.get("docname")
            mycursor=mydb.cursor()
            mycursor.execute('SELECT * FROM schedule WHERE docname=%s',(docname,))
```

```
            myresult=mycursor.fetchall()
            for x in myresult:
                print(x)
            mydb.commit()
        except:
            render_template('viewsch.html',user=user)
    try:
        return render_template('viewsch.html',data=myresult,user=user)
    except:
        return render_template('viewsch.html',  user=user)
@app.route('/doctor')
def doctor():
    if(g.user):
        return render_template('doctor1.html',user=session['user'])
    return render_template('login.html')
@app.route("/viewrecd",methods=['POST','GET'])
def view2():
    if 'user' in session :
        user = session['user']
    if (request.method == 'POST') :
```

```
        try:
            email=request.form.get("email")
            mycursor=mydb.cursor()
            mycursor.execute('SELECT * FROM addreco WHERE email=%s',(email,))
            myresult=mycursor.fetchall()
            for x in myresult:
                print(x)
            mydb.commit()
        except:
            render_template('viewd.html',user=user)
    try:
        return render_template('viewd.html',data=myresult,user=user)
    except:
        return render_template('viewd.html',  user=user)
@app.route("/editrecd",methods=['POST','GET'])
def edit1():
    if 'user' in session :
        user = session['user']
    if (request.method == 'POST') :


        try:
```

```
            email=request.form.get("email")
            mycursor=mydb.cursor()
            mycursor.execute('SELECT * FROM addreco WHERE email=%s',(email,))
            myresult=mycursor.fetchall()
            for x in myresult:
                print(x)
            mydb.commit()
        except:
            render_template('editd.html',user=user)
    try:
        return render_template('editd.html',data=myresult,user=user)
    except:
        return render_template('editd.html',  user=user)
@app.route('/update1')
def upd1():
    if(g.user):
        return render_template('updated.html',user=session['user'])
    return render_template('updated.html')
@app.route('/updaterecd', methods=['POST', 'GET'])
def update1():
    if request.method=='POST':
```

```
        name=request.form.get('name')

        email=request.form.get('email')

        dob=request.form.get('dob')

        gender=request.form.get('gender')

        contact=request.form.get('contact')

        medical=request.form.get('medical')

        cur=mydb.cursor()

        cur.execute(""" UPDATE addreco SET name=%s, email=%s, dob=%s, gender=%s,
contact=%s, medical=%s """,(name,email,dob,gender,contact,medical))

        flash("DATA UPDATED SUCCESSFULLY")

        mydb.commit()

   return render_template('editd.html')

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

def addsc():

   if 'user' in session:

      user = session['user']

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

      docname=request.form.get('docname')

      day=request.form.get('day')

      time=request.form.get('time')

      query = "insert into  schedule(docname,day,time) VALUES (%s,%s,%s)"

      val = (docname,day,time)
```

```
        mycursor = mydb.cursor()

        mycursor.execute(query, val)

        mydb.commit()

    return render_template('addsch.html')

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

def viewapp():

    if 'user' in session :

        user = session['user']

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



        try:

            docname=request.form.get("docname")

            mycursor=mydb.cursor()

            mycursor.execute('SELECT * FROM appointment WHERE
docname=%s',(docname,))

            myresult=mycursor.fetchall()

            for x in myresult:

                print(x)

            mydb.commit()

        except:

            render_template('viewap.html',user=user)

    try:
```

```
        return render_template('viewap.html',data=myresult,user=user)
    except:
        return render_template('viewap.html',  user=user)
@app.route('/patient')
def patient():
    if(g.user):
        return render_template('patient1.html',user=session['user'])
    return render_template('login.html')
@app.route("/viewrecp",methods=['POST','GET'])
def view1():
    if 'user' in session :
        user = session['user']
    if (request.method == 'POST') :


        try:
            email=request.form.get("email")
            mycursor=mydb.cursor()
            mycursor.execute('SELECT * FROM addreco WHERE email=%s',(email,))
            myresult=mycursor.fetchall()
            for x in myresult:
                print(x)
```

```
            mydb.commit()
        except:
            render_template('viewp.html',user=user)
    try:
        return render_template('viewp.html',data=myresult,user=user)
    except:
        return render_template('viewp.html',  user=user)
@app.route("/viewschp",methods=['POST','GET'])
def viewschp():
    if 'user' in session :
        user = session['user']
    if (request.method == 'POST') :


        try:
            docname=request.form.get("docname")
            mycursor=mydb.cursor()
            mycursor.execute('SELECT * FROM schedule WHERE docname=%s',(docname,))
            myresult=mycursor.fetchall()
            for x in myresult:
                print(x)
            mydb.commit()
```

```
        except:
            render_template('viewps.html',user=user)
    try:
        return render_template('viewps.html',data=myresult,user=user)
    except:
        return render_template('viewps.html',  user=user)
@app.route('/bookapp', methods=['POST','GET'])
def bookapp():
    if 'user' in session:
        user = session['user']
    if (request.method == 'POST') :
        name=request.form.get('name')
        emailid=request.form.get('emailid')
        docname=request.form.get('docname')
        day=request.form.get('day')
        time=request.form.get('time')
        print("1")
        query = "insert into  appointment(name,emailid,docname,day,time) VALUES
(%s,%s,%s,%s,%s)"
        val = (name,emailid, docname, day, time)
        mycursor = mydb.cursor()
        mycursor.execute(query, val)
```

```
        print("2")
        mydb.commit()
    return render_template('book.html')
@app.route("/logout")
def logout():
    session.pop('user', None)
    return render_template('login.html')
app.run(debug=True)
```

DATABASE:-

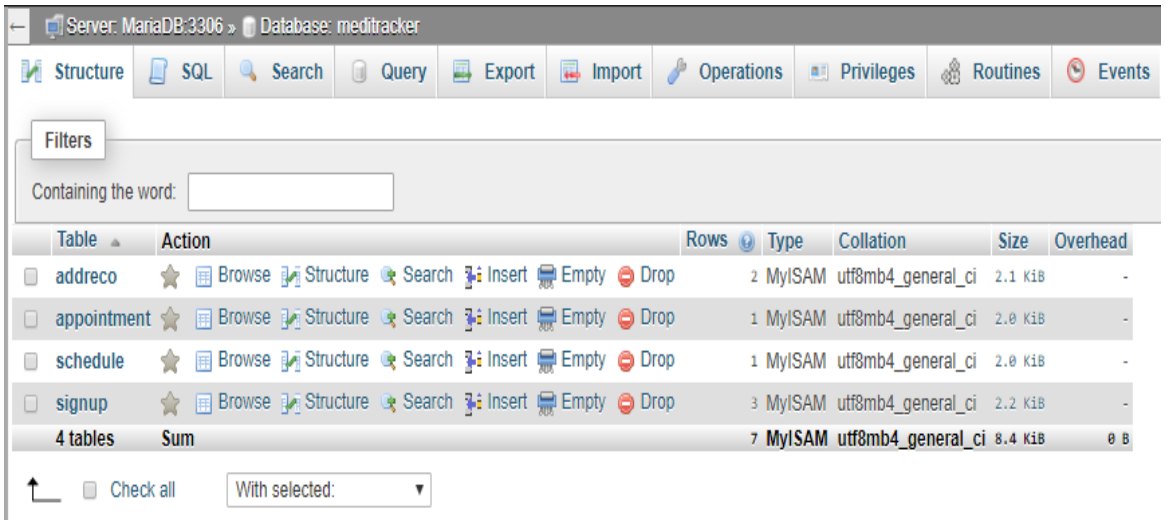


Fig 3.4.1 Meditracker Database used in this project

The Meditracker database used in this project consists of 4 tables. They are:

a) signup- To store the user details and login credentials
b) addreco- To store patient details
c) schedule- To store doctor availability schedule
d) appointments- To store the appointment booked by patient

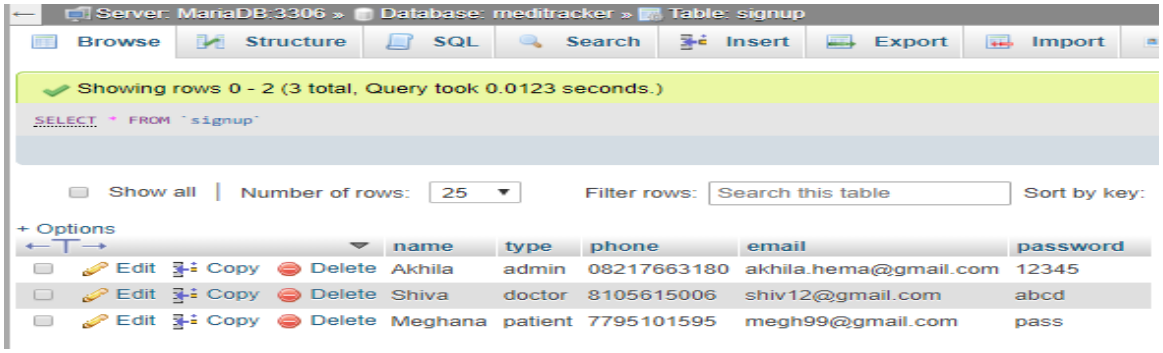


Fig 3.4.2 signup table

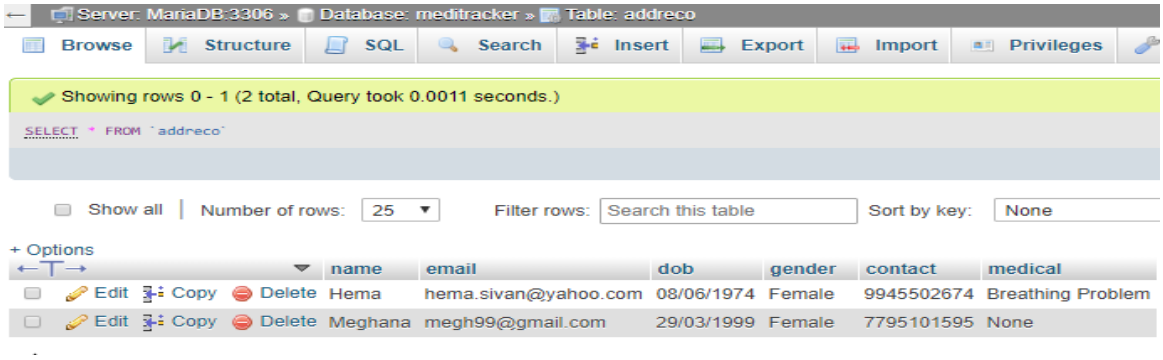


Fig 3.4.3 addreco table

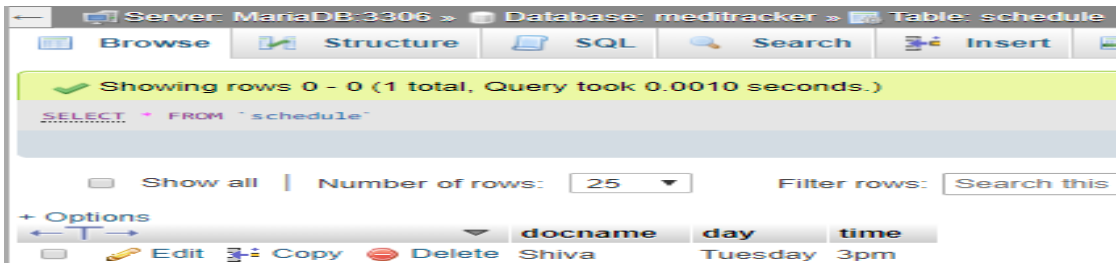


Fig 3.4.4 schedule table

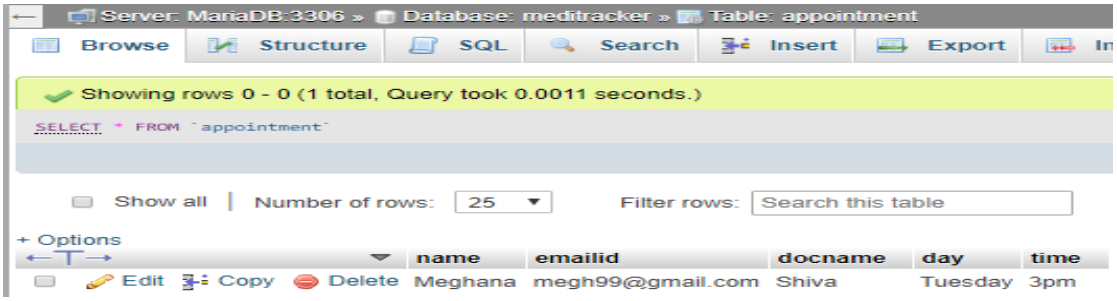


Fig 3.4.5 appointment table

Chapter 4

RESULTS AND DISCUSSION

4.1 Summary of result obtained

The outcome of this project is various types of user are able to perform different functions to get a desired output.

a) For the admin:
   - Add Record
   - View Record
   - Edit Record
   - View Schedule

b) For the doctor:
   - View Record
   - Edit Record
   - Add Schedule
   - View appointments

c) For the patient:
   - View Record
   - View Schedule
   - Book Appointment

4.2 Outputs (Snapshots)

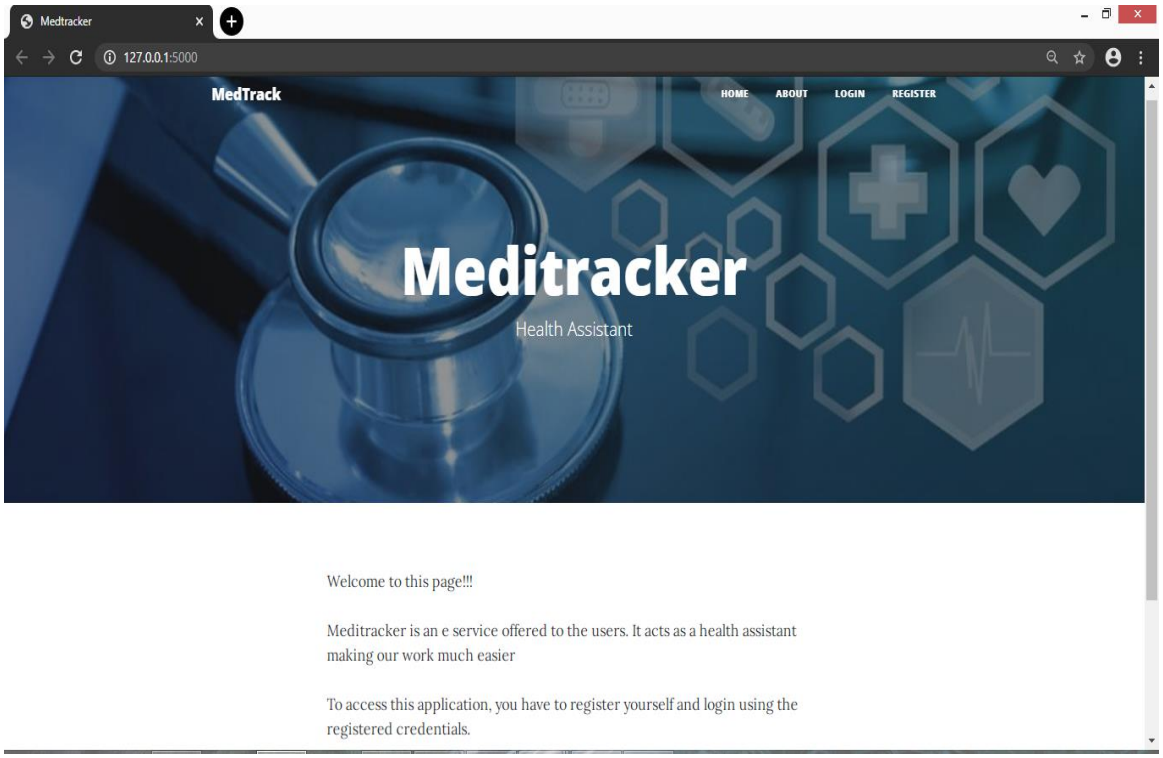


Fig 4.2.1 Homepage of the website

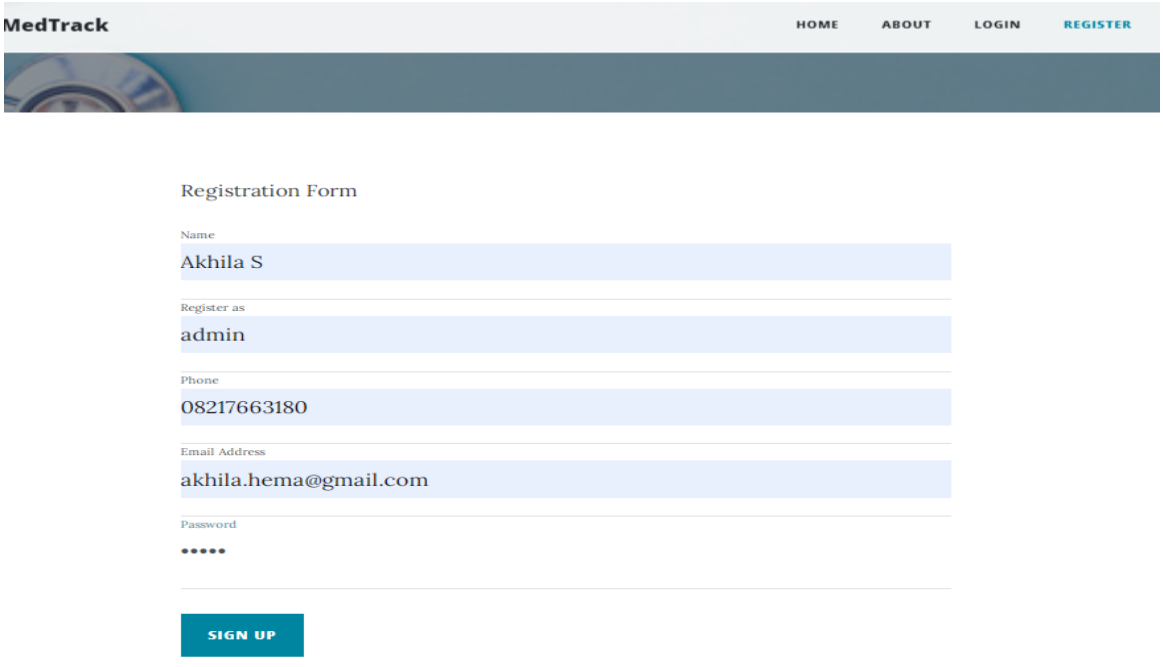


Fig 4.2.2 Registration page for different users

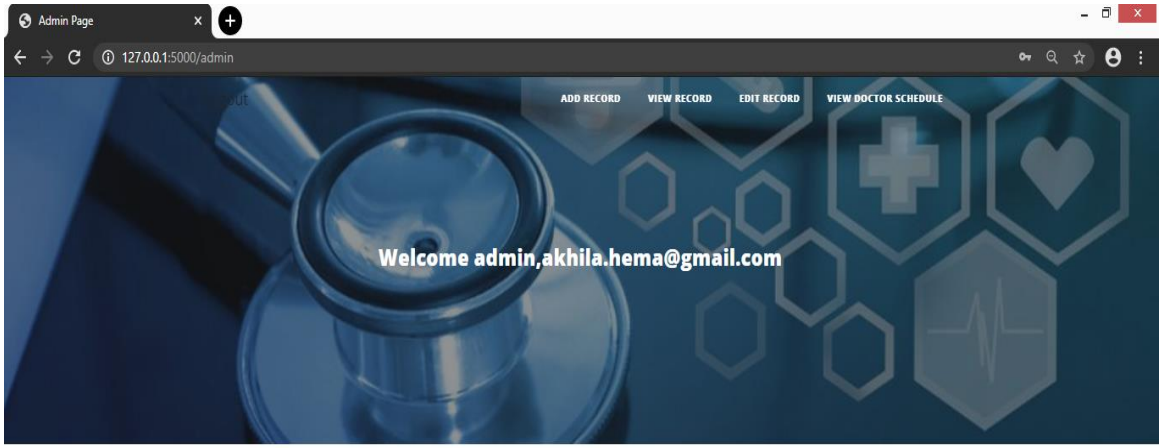


Fig 4.2.3 Admin Page after logging in as admin

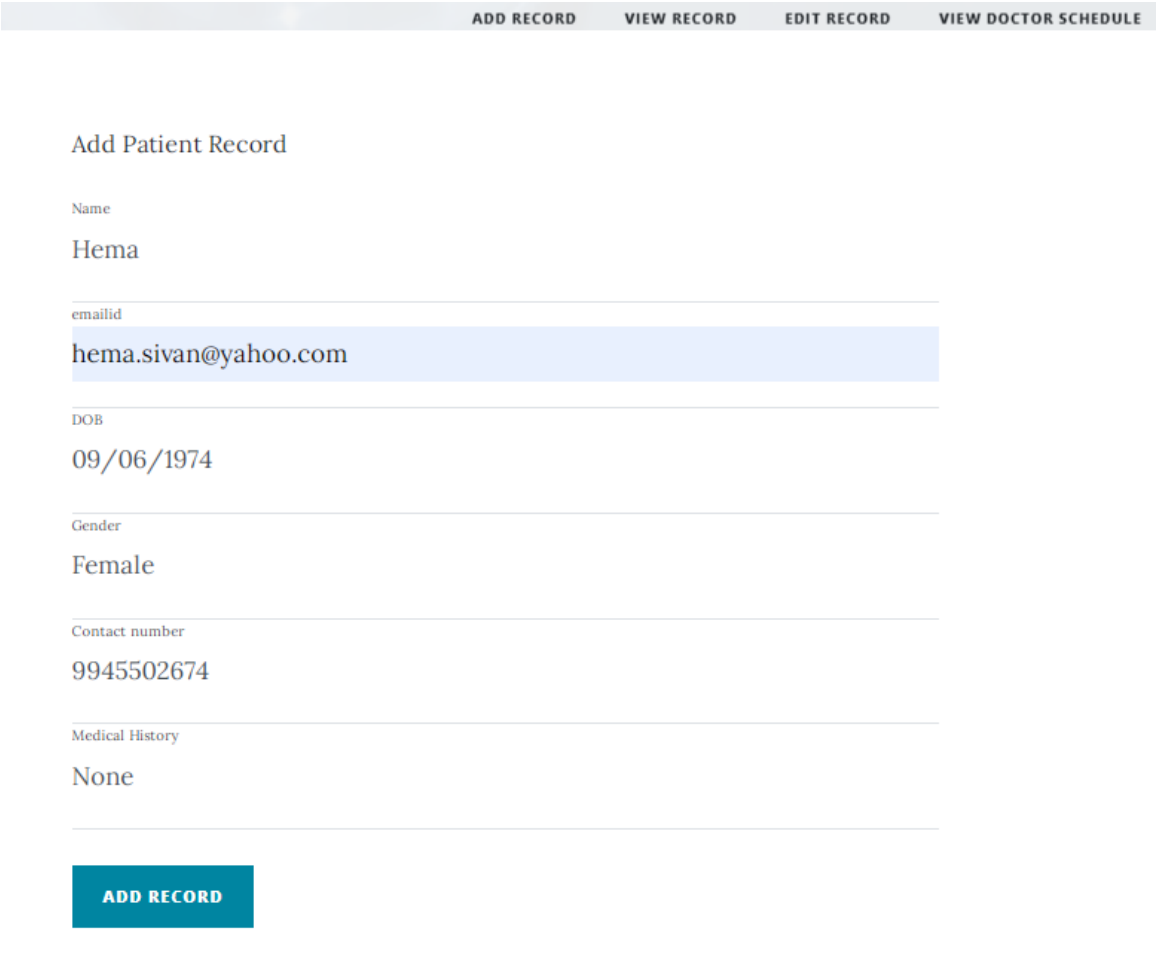


Fig 4.2.4 Admin- Add Patient Record

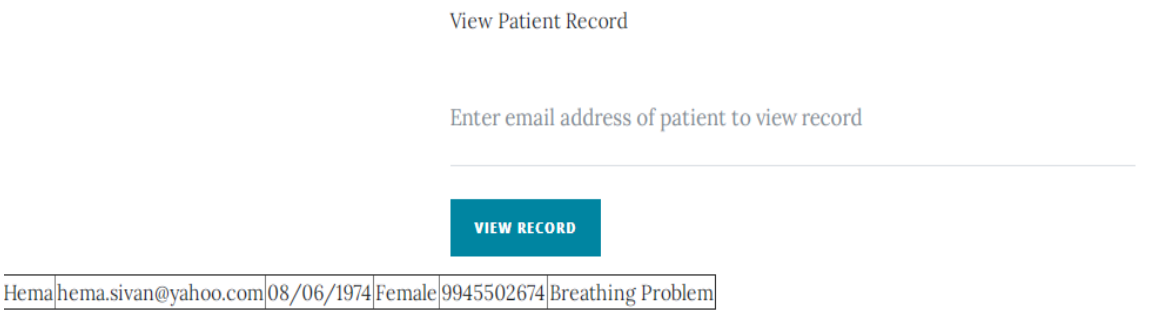


Fig 4.2.5 Admin- View Patient Record

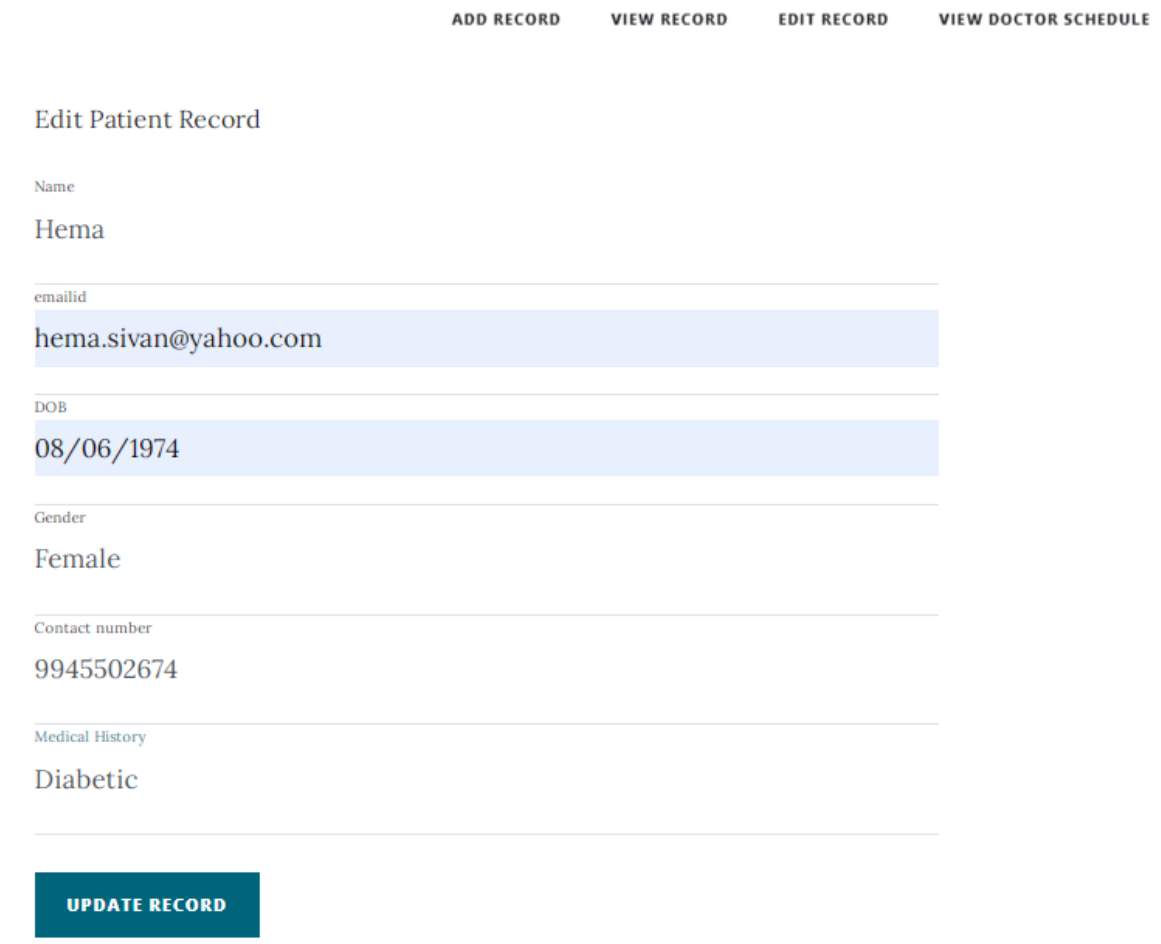


Fig 4.2.6 Admin- To Edit Patient Record

View Doctor Schedule

Enter doctor name to view schedule

VIEW SCHEDULE

Shiva	Tuesday	3pm

Fig 4.2.7 Admin- View Doctor Schedule

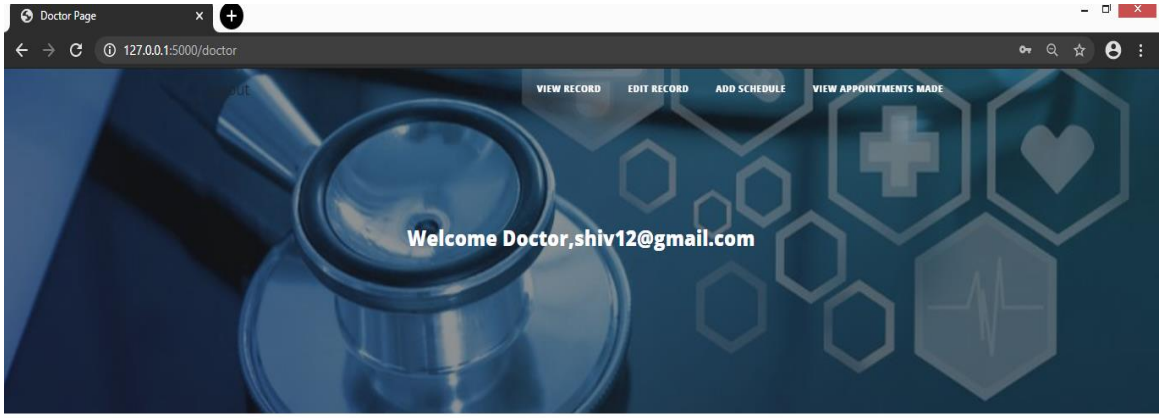


Fig 4.2.8 Doctor Page after logging in as a doctor user

View Patient Record

Enter email address of patient to view record

VIEW RECORD

Hema	hema.sivan@yahoo.com	08/06/1974	Female	9945502674	Breathing Problem

Fig 4.2.9 Doctor: View Patient Record

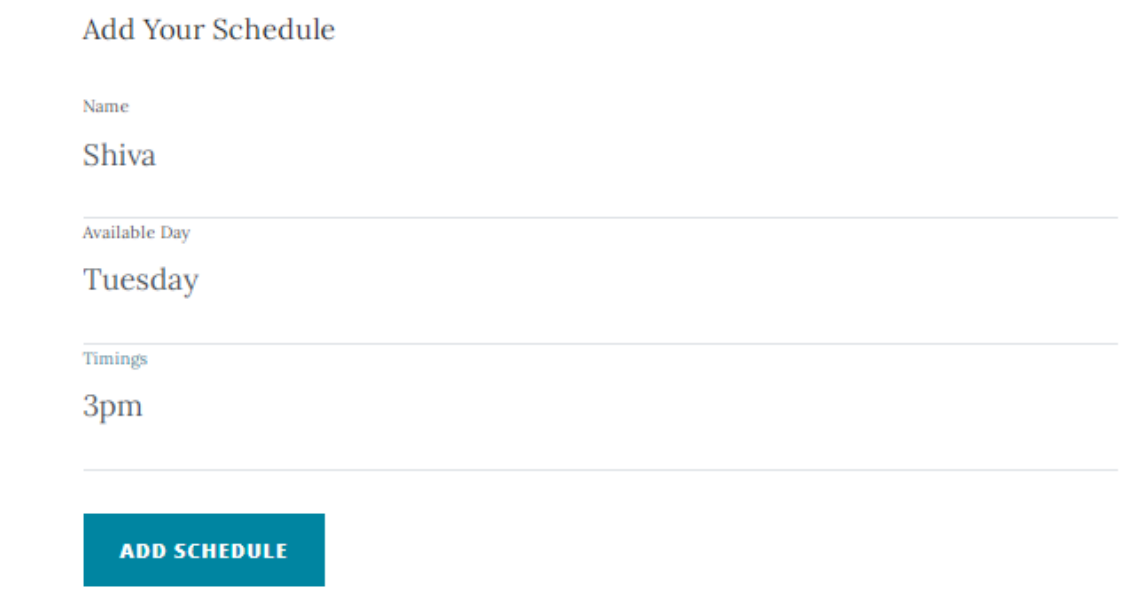


Fig 4.2.10 Doctor: Add Schedule

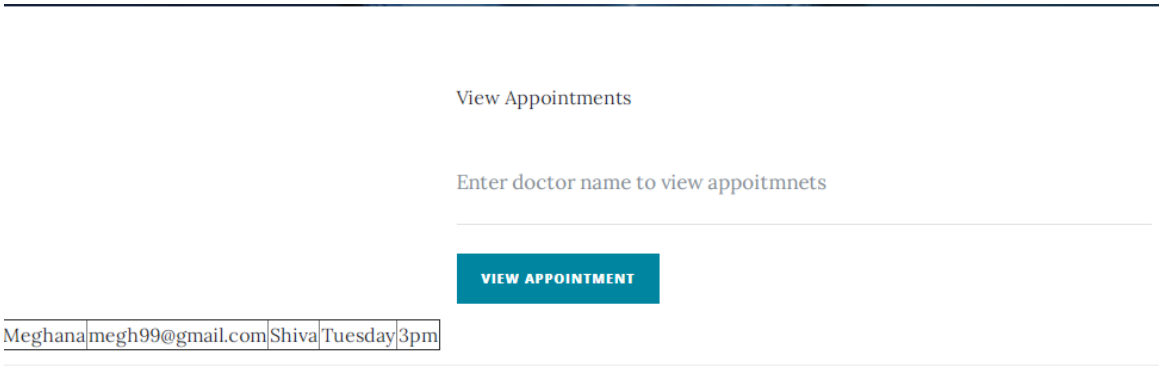


Fig 4.2.11 Doctor: View appointments

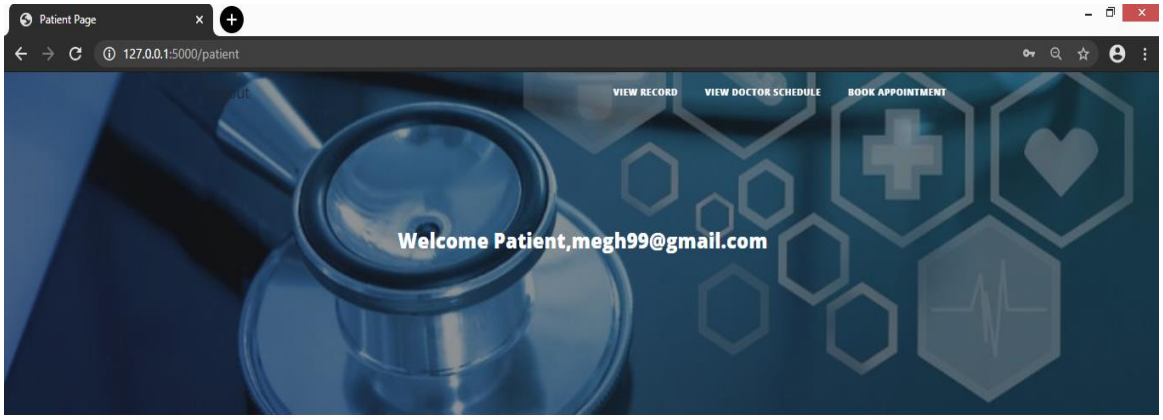


Fig 4.2.12 Patient page after logging in as patient user

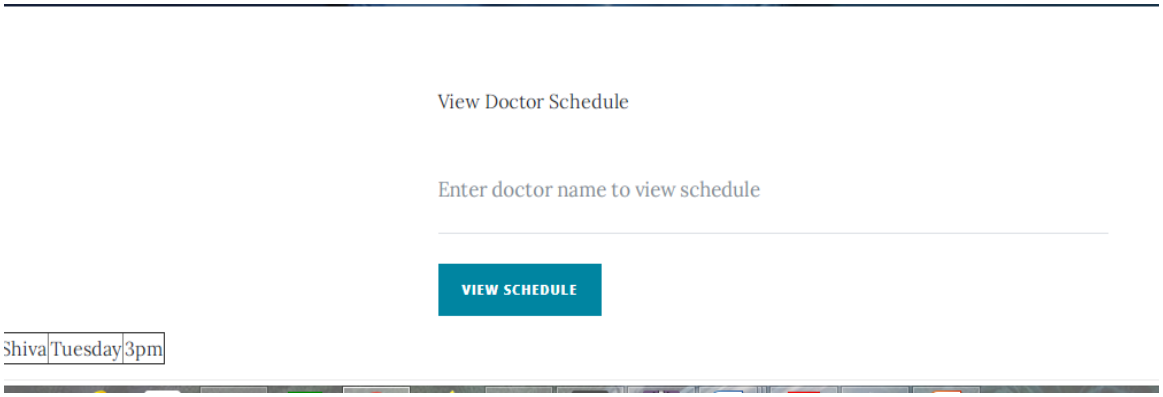


Fig 4.2.13 Patient: View Doctor Schedule

Fig 4.2.14 Patient: View Record

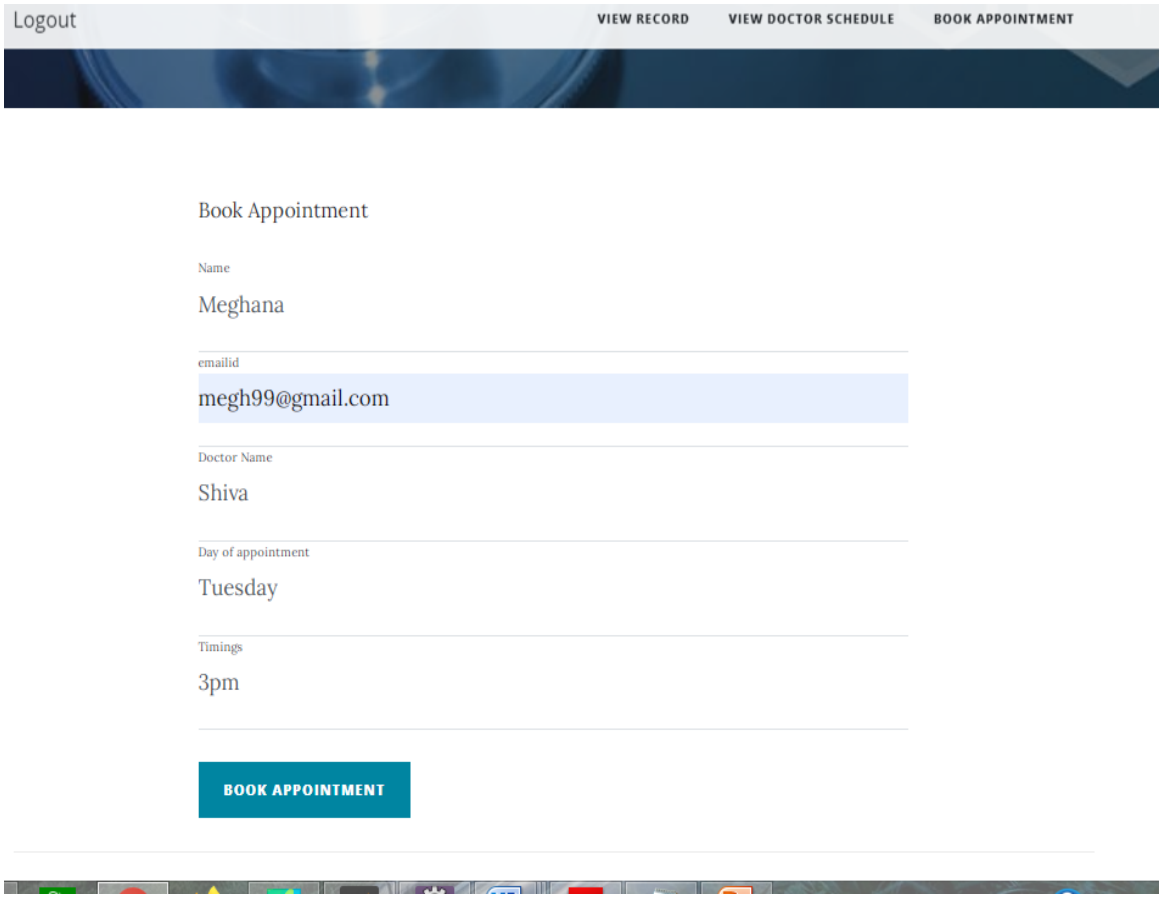


Fig 4.2.15 Patient: Book Appointment

Chapter 5:

CONCLUSION

Meditracker acts as an e-health assistant where the different types of users can avail the functionalities provided for them with much ease. The proposed system is much adaptable because it is user friendly as the website is designed in much simple manner where the user has to just click on buttons or navigation link to perform some actions. Also this application will reduce the manual work and improve the quality of maintaining records and other information related. Information retrieval is also easy as the data is in stored in tables. Hence this website created using HTML, CSS, and JavaScript with backend as MYSQL database is the most preferred solution.

REFERENCES

1) Python programming: CS (3rd edition)

-John Zelle

2) Fundamentals of Web Development

-Randy Connolly, Ricardo Hoar

3) Flask Documentation