**A Project Report on**

**Mhealth-Get Healthier**

Submitted by

**Saloni Redij (18IT1047)**

**Shreya Satav (18IT1012)**

**Manthan Shah (18IT2003)**

**Under the guidance of**

**Mrs. Anita Senathi Assistant Professor**

**Department of Information Technology Ramrao Adik Institute of Technology**

**Nerul, Navi Mumbai**

**Certificate**

**This is to certify that the project entitled Leave Management System is being submitted to the Department of Information Technology, Ramrao Adik Institute of Technology, Navi Mumbai.**

**Project Guide External Examiner(Mrs. Anita Senathi) ( )**

**Acknowledgment**

**We owe our gratitude to many people who have supported us throughout this journey.**

**We would firstly like to express our heartfelt gratitude towards our respected Principal Dr. Ramesh Vasappanavara and our Head of Department Dr. Ashish Jadhav for providing us immense facilities, guidance, and never-ending support.**

**The completion of any inter-disciplinary project depends upon cooperation and combined efforts of several sources of knowledge. We take this opportunity to express our profound gratitude and deep regard to our guide Mrs. Anita Senathi for her exemplary guidance, monitoring and constant encouragement throughout the course of this project.**

**Lastly, we thank our parents, family, friends, and well-wishers who always looked for the chance to help us in whatever means come forth and for their constant encouragement without which the project would not be a distant reality.**

**Proposed System**

Right now many people face the problem of lack of availability of doctors in our nation. Due to this, they are not able to get proper treatment at the right time.

The main objective of the proposed system is to solve the problem of the lack of availability of doctors. Here the user has to create an account and then the user can check BMI(**Body Mass Index**) and the disease the user might be suffering from and treat the patient accordingly.

**System Components**

**1.Frontend**: HTML5,CSS

**2.Backend**: Python, Django framework 2.0.3

**3.Database**: Dbsqlite3

**1.1 HTML:**

HTML is HyperText Mark-up Language used for documents (called pages) that are displayed on the World Wide Web. Each page contains a series of connections to other pages called hyperlinks. Every web page you see on the Internet is written using one version of HTML code or another.HTML code ensures the proper formatting of text and images so that your Internet browser may display them as they are intended to look. Without HTML, a browser would not know how to display text as elements or load images or other elements. HTML also provides a basic structure of the page, upon which Cascading Style Sheets are overlaid to change its appearance. The user interface is designed using HTML CSS and Bootstrap.

**1.2 CSS:**

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

**1.3 JAVASCRIPT:**

JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

**2.1Django(2.0.3):**

Django is a free and open-source web framework, written in Python, which follows the model-view-template architectural pattern. In our project, we have used this framework to host our leave management system in which users will log in to apply for leave that will be approved or not approved by admin.

Django is a widely-used Python web application framework with a "batteries-included" philosophy. The principle behind batteries-included is that the common functionality for building web applications should come with the framework instead of as separate libraries. Authentication, URL routing, a template engine, an object-relational mapper (ORM), and database schema migrations are all included with the Django framework. Compare that included functionality to the Flask framework which requires a separate library such as Flask-Login to perform user authentication.

Companies, organizations, and governments have used Django to build all sorts of things — from content management systems to social networks to scientific computing platforms.

**2.3 Python:**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a

scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed. The fast edit-test-debug cycle makes this simple approach very effective.

**3.1 Dbsqlite3:**

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects. Dbsqlite3 is a built-in database provided Django framework.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file.

**Architecture**

**Mhealth (Use case Diagram)**

WORKING OF SYSTEM

The Mhealth basically consist of one module:-

**1. User**

The user member logs into his/her interface window where he can check:

* Personal BMI
* Disease the user might be suffering from
* Can check for some other member through the account

When the user logs into the system, the user can check its BMI as well as check the disease the user might be suffering from.

In order to check the BMI:

The user is asked to enter its weight and height and according to it the BMI is calculated.

In order to check the Disease:

The user is asked to select the symptoms and the disease is displayed according to the symptoms the user has selected.

**Advantages of Mhealth**

1.Calculate the BMI without the need of a doctor

2. Can have a rough idea about the disease the user is suffering from

3. NO need to wait for the doctor.

**1.Views.py(Accounts)**

from django.shortcuts import render,redirect

from django.contrib import messages

from django.contrib.auth.models import User,auth

def login(request):

if request.method == "POST":

username = request.POST['username']

password = request.POST['password']

user = auth.authenticate(username=username,password=password)

if user is not None:

auth.login(request,user)

return redirect('/')

else:

messages.info(request, 'Invalid Credentials')

return redirect('login')

else:

return render(request, 'login.html')

def logout(request):

auth.logout(request)

return redirect('/')

def register(request):

if request.method == 'POST':

first\_name = request.POST['first\_name']

last\_name = request.POST['last\_name']

username = request.POST['username']

password1 = request.POST['password1']

password2 = request.POST['password2']

email = request.POST['email']

if password1 == password2:

if User.objects.filter(username=username).exists():

messages.info(request,'Username unavailable')

return redirect('register')

elif User.objects.filter(email=email).exists():

messages.info(request,'Email already exists')

return redirect('register')

else:

user = User.objects.create\_user(username=username,password=password1,email=email,first\_name=first\_name,last\_name=last\_name)

user.save()

messages.info(request,'User Created')

else:

messages.info(request,"password does not match")

return redirect('register')

return redirect('login')

else:

return render(request,'register.html')

**2.Views.py(Mhealthapp)**

from django.shortcuts import render,redirect

from django.contrib import messages

# Create your views here.

def index(request):

return render(request, 'index.html')

def bmi(request):

return render(request, 'bmi.html')

def symptom(request):

if request.method == "POST":

cough = request.POST['cough']

fever = request.POST['fever']

common = request.POST['common']

if(cough == 'not' and fever == 'not' and common == 'not'):

messages.info(request, 'You are healthy. Exercise daily and stay fit.')

return redirect('symptom')

elif (cough == 'candc' and fever == 'not' and common == 'not'):

messages.info(request,'Common cold and Cough')

return redirect('symptom')

elif ((cough == 'candc' or cough == 'not' or cough == 'drysore') and (fever == 'not' or fever == 'highfever' or fever == 'normalfever') and common == 'diarrheoa' ):

messages.info(request, 'Food Poisoning')

return redirect('symptom')

elif (cough == 'sputum'):

messages.info(request, 'Tuberculosis')

return redirect('symptom')

elif (cough != 'sputum' and fever == 'pattern' and common == 'diarrheoa'):

messages.info(request, 'Typhoid')

return redirect('symptom')

elif (cough == 'sore throat' and (fever == ' not' or fever == 'normalfever') and common == 'not'):

messages.info(request, 'Throat Infection')

return redirect('symptom')

elif (cough != 'sputum' and (common == 'yellow' or common == 'all')):

messages.info(request, 'Jaundice')

return redirect('symptom')

elif (cough == 'candc' and (fever == 'normalfever' or fever == 'highfever') and common == 'diarrheoa'):

messages.info(request, 'Viral')

return redirect('symptom')

else:

messages.info(request, 'Uh-oh!Disease not found.')

return redirect('symptom')

else:

return render(request, 'symptom.html')

**Javascript**

**3.bmid.js**

function myBmi() {

var a,g,w,h,bmi\_result,bmi\_child;

a = document.getElementById("form1");

g = a.elements["age"].value;

h = parseFloat(a.elements["height"].value);

w = parseFloat(a.elements["weight"].value);

bmi\_result = w / (h\*\*2) ;

bmi\_child = w / (h\*\*2) \* 703;

**4.accounts.js**

function alertBox(){

alert({{message}});

}

if (g == 'adult'){

document.getElementById("demo1").innerHTML="Your BMI is: " +bmi\_result;

if (bmi\_result < 18.5 ) {

document.getElementById("demo").innerHTML="Underweight";

}

else if(bmi\_result >= 18.5 && bmi\_result <=24.9 ) {

document.getElementById("demo").innerHTML="Normal Weight";

}

else if(bmi\_result >= 25 && bmi\_result <=29.9 ){

document.getElementById("demo").innerHTML="Over Weight";

}

else {

document.getElementById("demo").innerHTML="Obessed";

}

}

else{

document.getElementById("demo1").innerHTML="Your BMI is: " +bmi\_child +" percentile";

if (bmi\_child < 5 ){

document.getElementById("demo").innerHTML="Underweight";

}

else if(bmi\_result >= 5 && bmi\_result < 85 ){

document.getElementById("demo").innerHTML="Normal Weight";

}

else if(bmi\_result >= 85 && bmi\_result < 95 ){

document.getElementById("demo").innerHTML="Over Weight";

}

else {

document.getElementById("demo").innerHTML="Obese";

}

}

}

**Django Source Code**

**2. Models.py(accounts)**

from django.db import models

# Create your models here.

class Register(models.Model):

name: models.CharField(max\_length=100)

**3. accounts/urls.py**

from django.urls import path

from . import views

urlpatterns = [

path("register/", views.register, name="register"),

path("login/", views.login, name="login"),

path("logout", views.logout, name="logout"),

]

**4. mhealthapp/urls.py**

from django.urls import path

from . import views

urlpatterns = [

path('',views.index, name='index'),

path('bmi/',views.bmi , name = 'bmi'),

path('symptom/',views.symptom , name = 'symptom'),

]

**4. mhealth/urls.py(Main module)**

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

path('admin/', admin.site.urls),

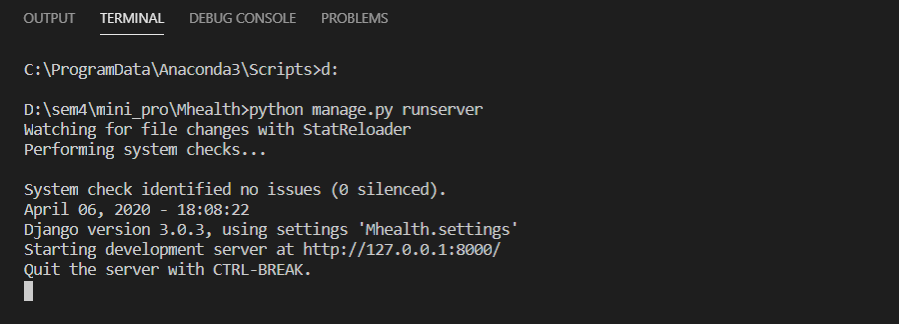
path('',include('Mhealthapp.urls')),

path('accounts/', include('accounts.urls')),

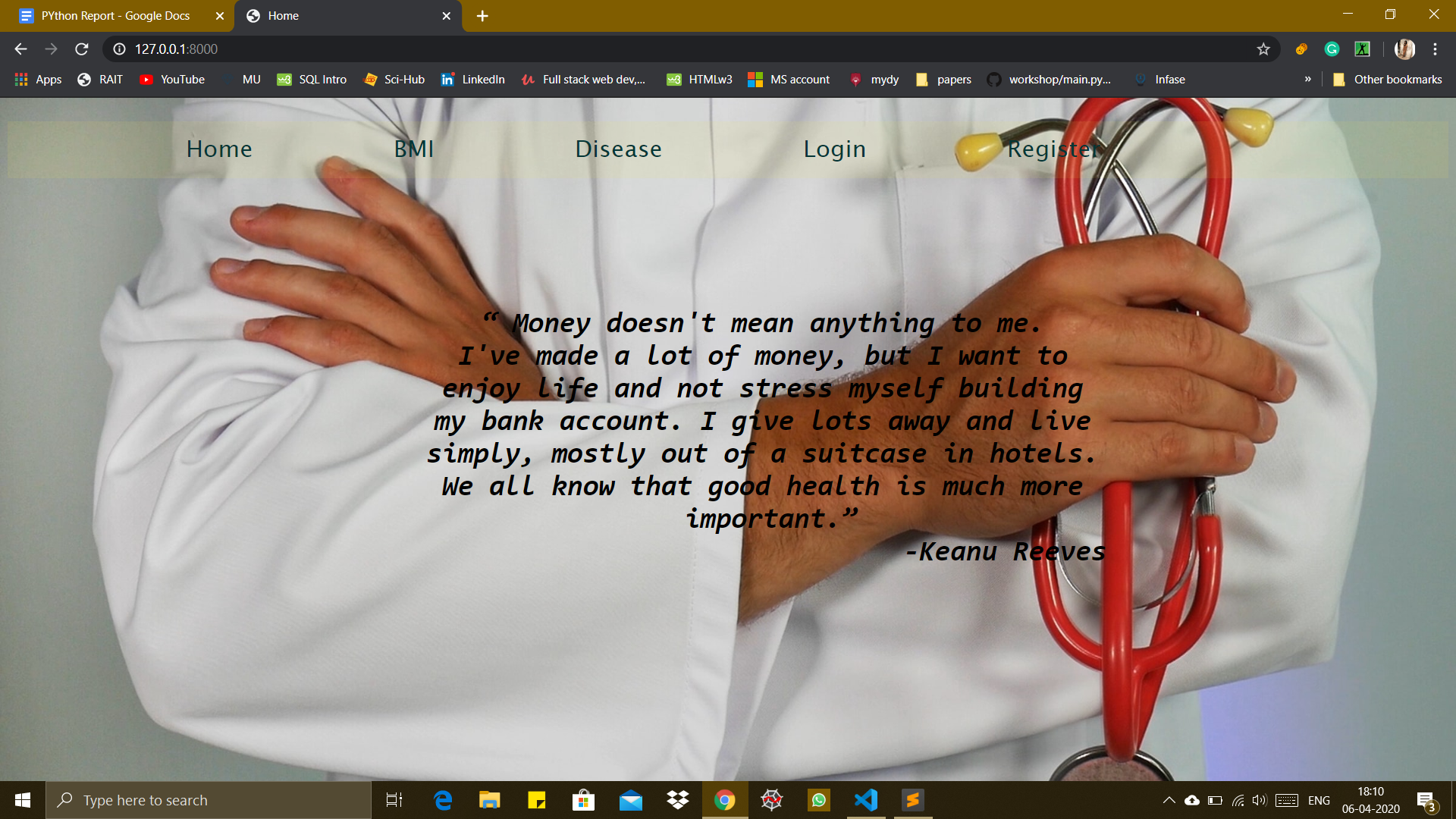
]

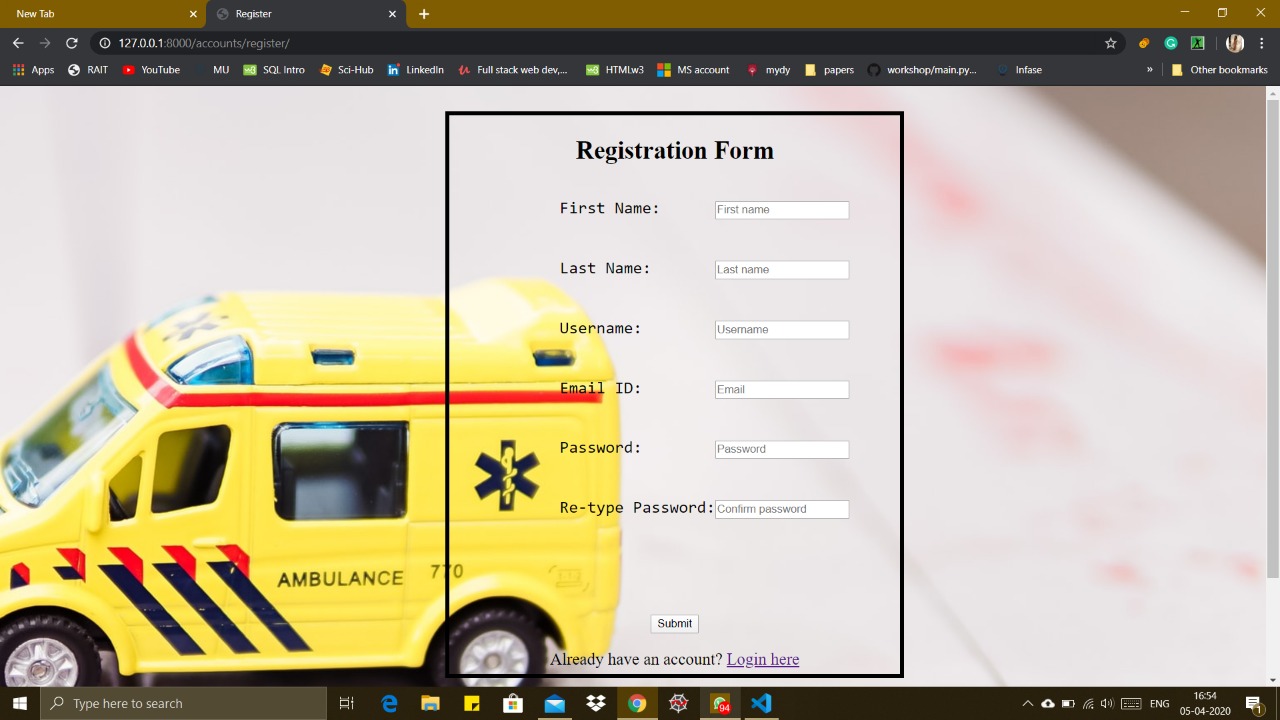
**Implementation of Web Application**

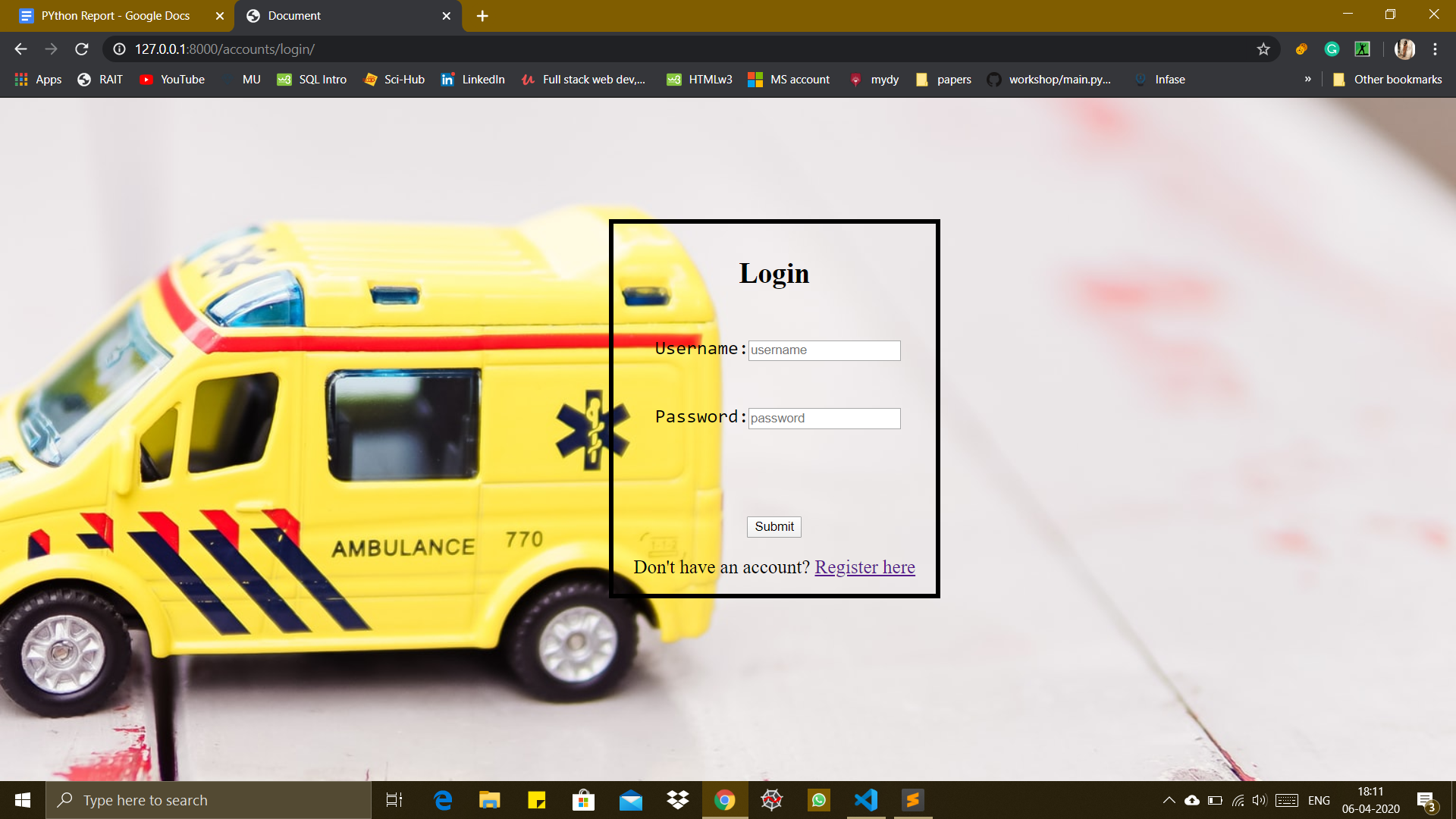
1. Start the Django Web Server.



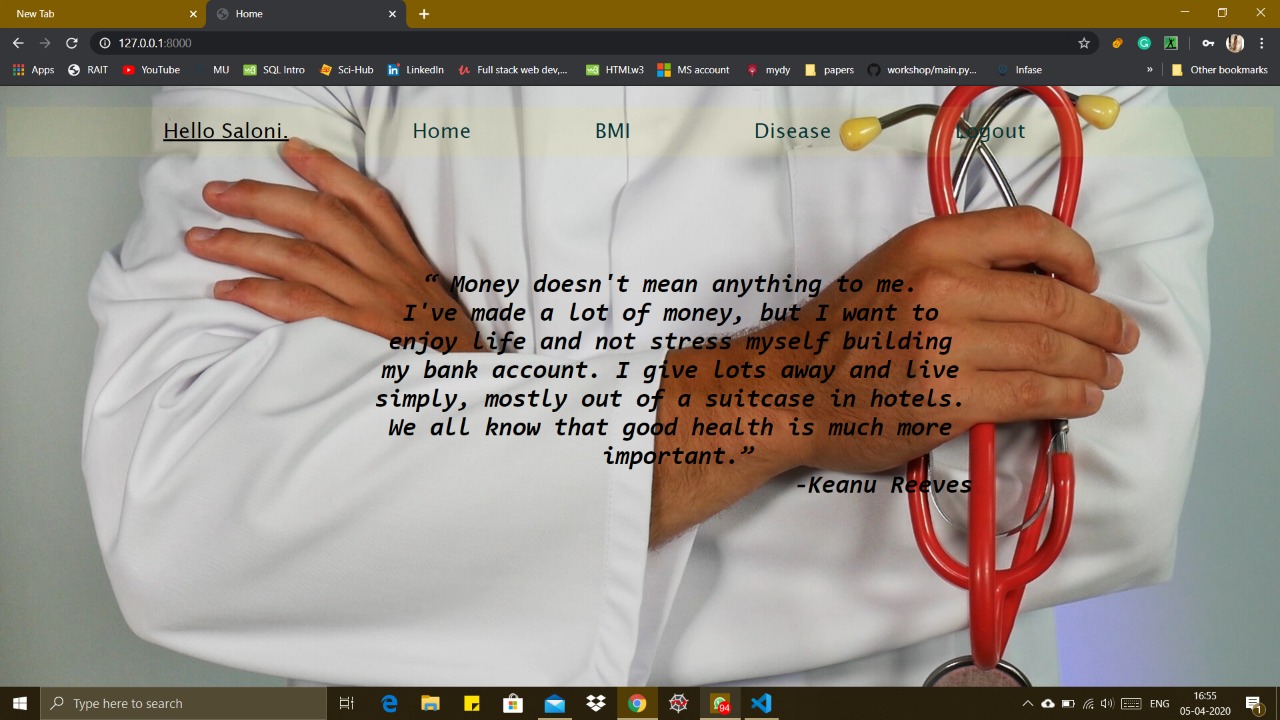
2. To check the server is running or not type localhost:8000 in the browser(using Google Chrome here). If the server is running the staring page of the website will be displayed.

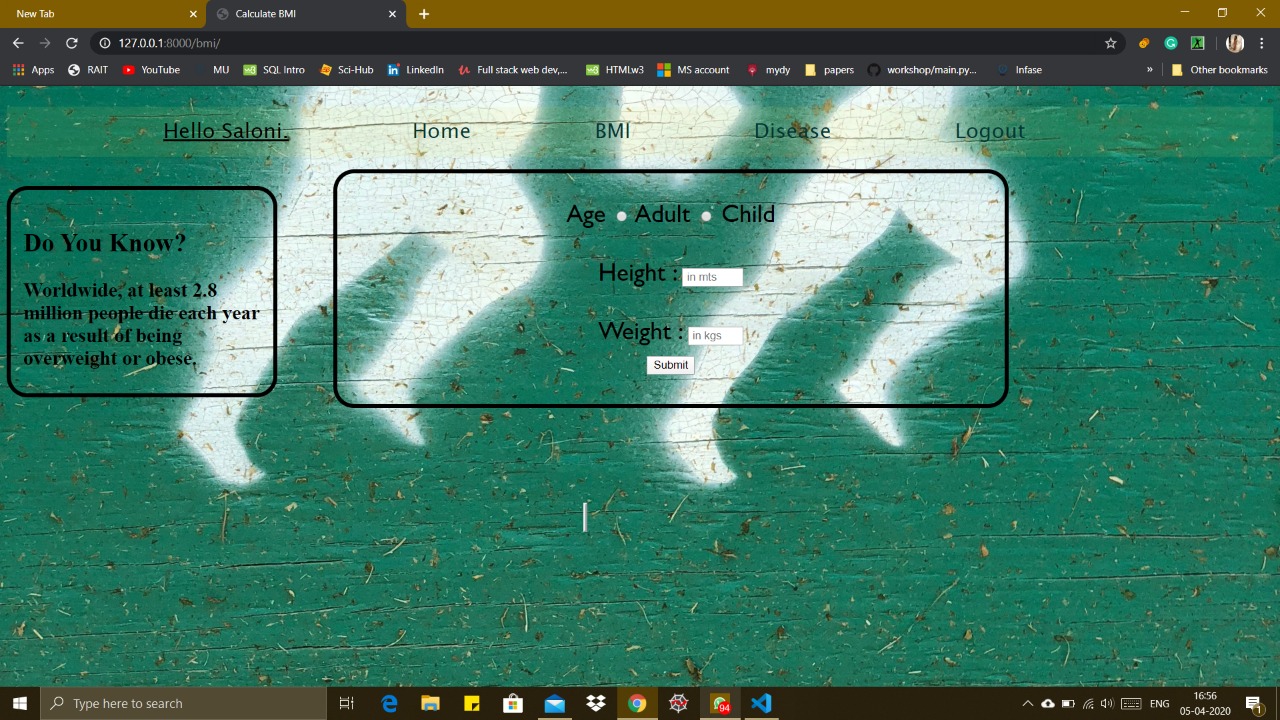
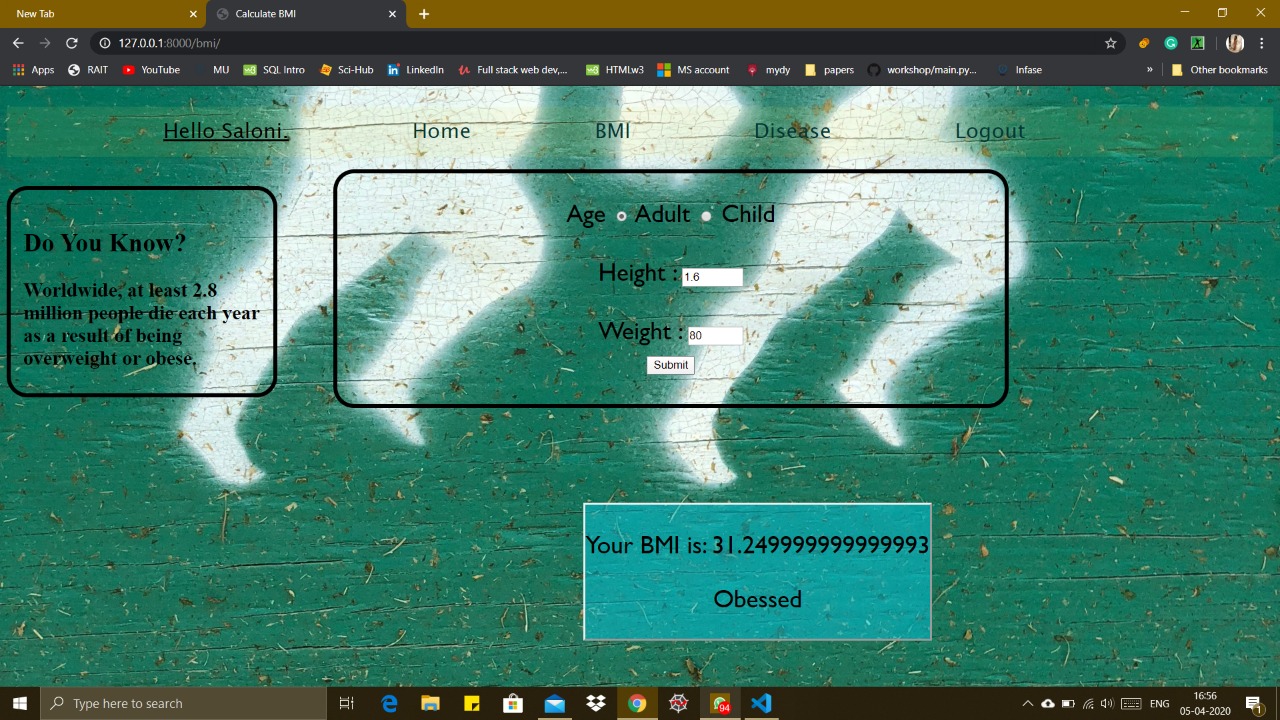
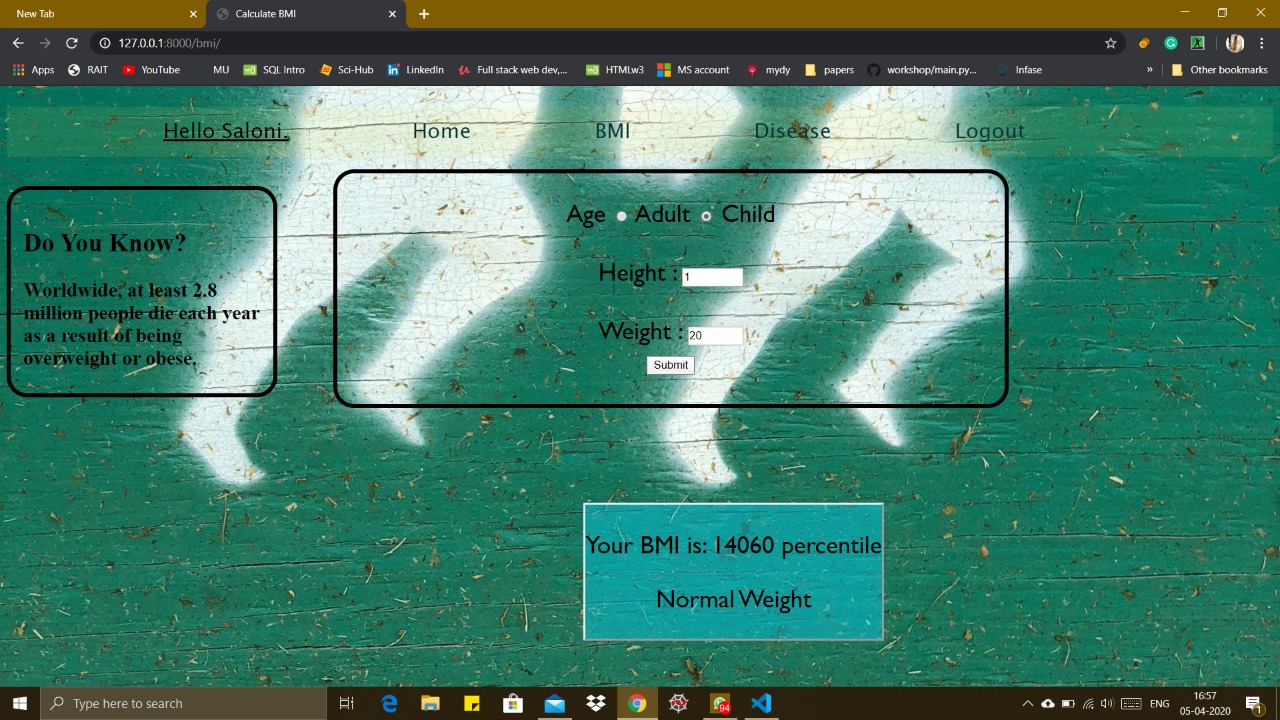


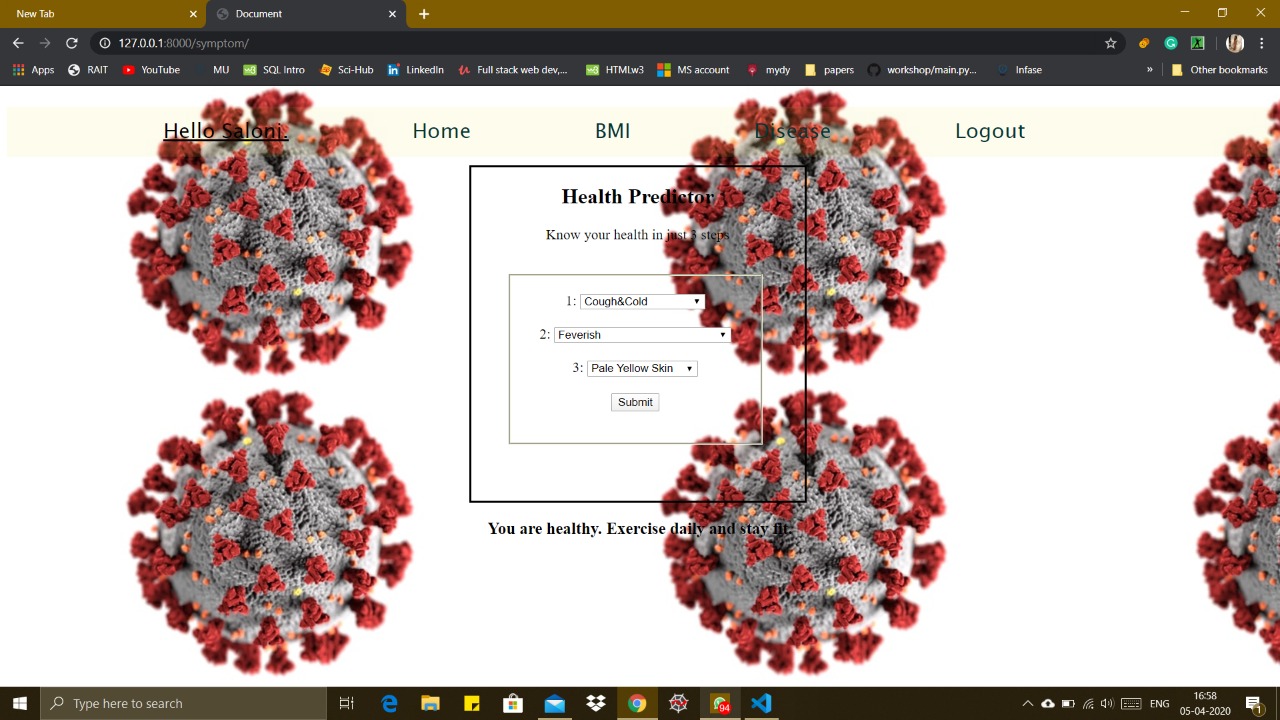
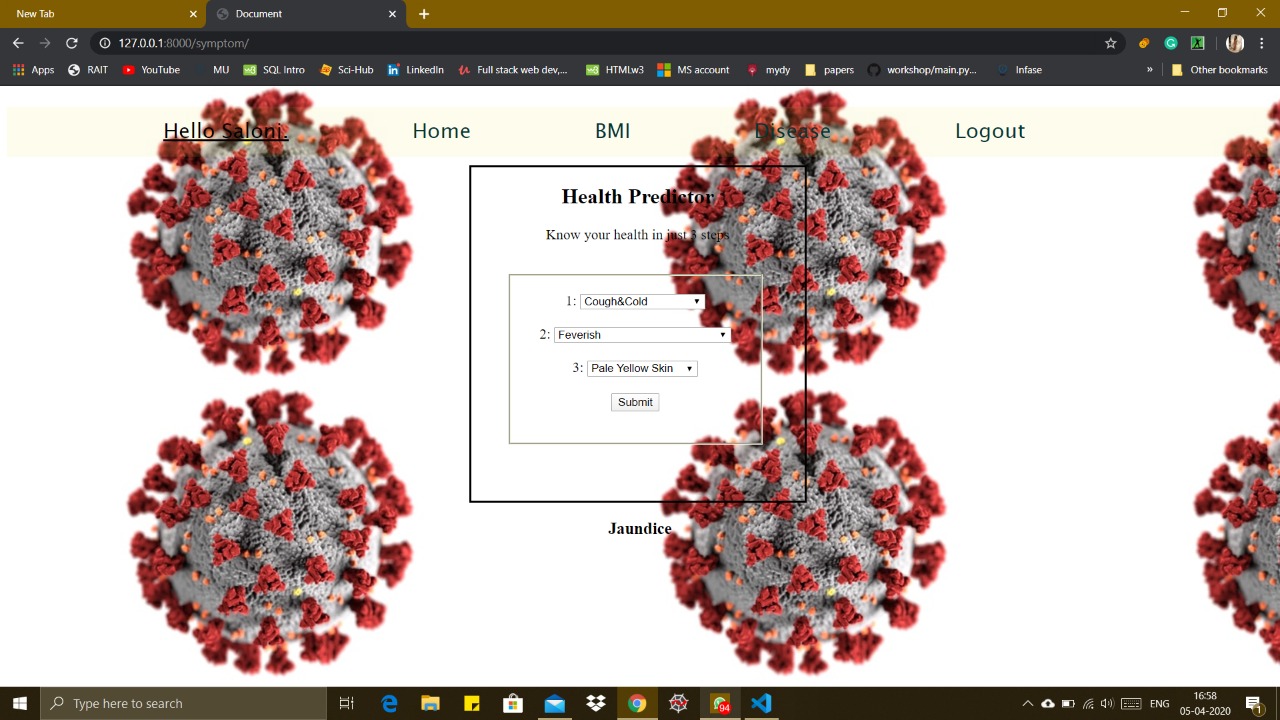
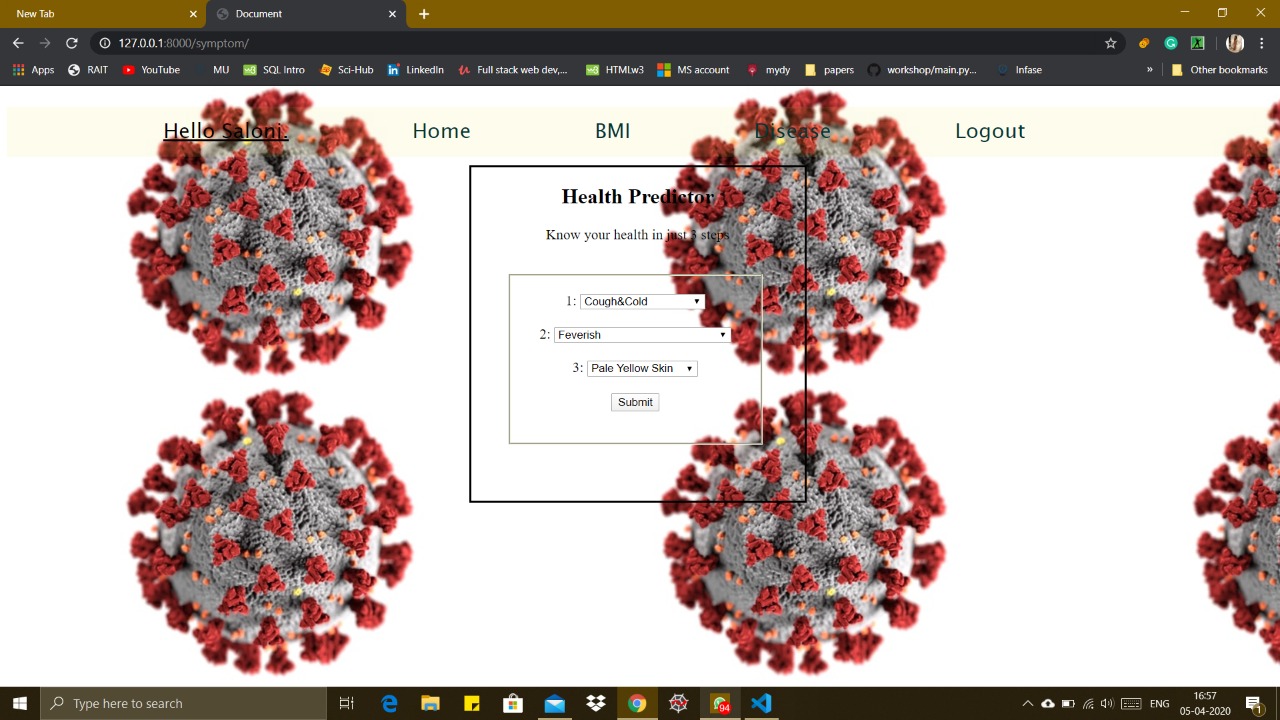




3. After selecting a login user will see this.



4. As a user tries to check its BMI

5. When a user wants to check the disease the user is suffering from

**Conclusion**

Mhealth lets the user check your BMI and disease the user might be suffering from.

**References**

* Django documentation (2.0.3)
* The Definitive Guide to Django Web Development Done Right by  
  Adrian Kaplan-Moss, Jacob
* YouTube channels- TechwithTim and Telusko
* Python Documentation(3.7)
* <https://www.w3schools.com/> for HTML, CSS, Javascript