A PROJECT REPORT ON

WE CARE

AN E-HEALTH CARE SYSTEM

By

Nirali Shah (CE122) (18CEUOS059)

Krishna Shah (CE064) (18CEUON117)

B. Tech CE Semester - VI

Subject: System Design Practice

Guided By:

Prof. Malay Bhatt

Associate Professor

Department Of Computer Engineering

Faculty of Technology

Department of Computer Engineering

Dharmsinh Desai University

Faculty of Technology

Department of Computer Engineering

Dharmsinh Desai University

CERTIFICATE

This is to certify that the practical / term work carried out in the subject of **System Design Practice** and recorded in this journal is the bonafide work of

KRISHNA SHAH (CE064) (18CEUON117)

NIRALI SHAH (CE122) (18CEUOS059)

of B.Tech semester **VI** in the branch of **Computer Engineering** during the academic year **2020-2021**.

Prof Malay S Bhatt
Associate Professor,
Dept. of Computer Engg.,
Faculty of Technology
Dharmsinh Desai University, Nadiad

Dr. C. K. Bhensdadia,
Head,
Dept. of Computer Engg.,
Faculty of Technology
Dharmsinh Desai University, Nadiad

Table of Contents

Introduction

1. Purpose

The purpose of We Care an E-Healthcare Expert System is to provide health related information at one platform. This system helps people to find health and medical information, support and services they need. This project is very helpful to both Medicare staff as well as to the public.

2. Need for the project

The growing quality demand in the hospital sector makes it necessary for such a system where one can find all required information. The system makes it easy for people to make an initial diagnosis of their disease and find the initial treatment. Also it is easy for doctors to find other doctors and take their consultations into consideration.

3. Scope

This system is web based application which provides valuable health related information to its users. One can find disease related information, its causes and its primary treatment from this site. Users can also find nearby hospitals and pharmacy. The system identifies all nearby hospital and pharmacies based on the user's current location. If any user has questions they can post it on the FAQ page of the system.

It is easy for doctors to find treatment for various diseases as everything is just a click away. They can talk to other doctors and take into consideration their point of view. Also they can post answers to questions raised by the users.

The medical news section provides with various medicine and health related news. Both users and the doctors have access to the news section of the system.

The system provides with all latest information regarding the Covid-19 pandemic. Data can be fetched for a particular country or all countries also.

The admin can generate reports regarding the most searched symptoms and the treatment.

4. Tools/Technologies Used

Technologies

- Python-version 3.8
- Sqlite Database

Framework

- Django Framework – version 3.2

Tools

- Visual Code

Platform

- Web Application with Python

System Requirements Specifications

1. Users

- Admin
- Health Professionals /Doctor
- Users/Patient

2. Functional Requirements

2.1 Manage Disease Information

2.1.1 Description and priorities

This feature is related to diseases information and used by all types of users.

2.1.2 Functional Requirements

2.1.2.1 Add new disease information

Input: Disease information like name, symptoms, treatment etc.

Output: Information is added successfully.

2.1.2.2 Update existing disease information

Input: changes needed

Output: Information is updated successfully.

2.1.2.3 Delete Existing Disease Information

Input: disease name

Output: Information is deleted successfully.

2.1.2.4 Fetch Diseases based on Symptoms

2.1.2.4.1 Fetch Disease by image

Input: user selection

Output: based on selection disease information is displayed

2.1.2.4.2 Fetch Disease by Symptoms

Input: various symptoms

Output: based on symptoms most likely matched disease information is displayed.

2.2 Manage Disease Treatment Details

2.2.1 Description and perspective

Primary disease treatment details are provided and advice for higher treatment if necessary.

2.2.2 Functional Requirement

2.2.2.1 Add treatment details

Input: Treatment details

Output: Details are added successfully

2.2.2.2 Modify treatment details

Input: Change needed

Output: Details are updated

2.2.2.3 Fetch treatment details

Input: Disease name, symptoms

Output: Related treatments are displayed

2.3 Manage Hospital Information

2.3.1 Description and perspective

Hospital information which is provided by hospital administration can be seen by user.

2.3.2 Functional Requirements

2.3.2.1 Fetch nearby hospital based on location

Input: Location

Output: based on location list of hospitals is displayed

2.3.2.2 Fetch hospital by name

Input: Name of hospital

Output: hospital details is displayed

2.3.2.3 Read about a hospital

Input: User selection

Output: Information can be viewed.

2.3.2.4 Add information about a hospital

Input: Hospital details

Output: Information related to hospital is added successfully.

2.3.2.5 Update hospital Information

Input: changes needed

Output: details are updated

2.4 Manage FAQ

2.4.1 Description and perspective

User can post question, view others answers. The professionals post answers to the questions.

2.4.2 Functional Requirements

2.4.2.1 Post question

Input: question

Output: question is posted successfully

2.4.2.2 Answer question

Input: answer related to question

Output: answer is added can be seen by all users

2.4.2.3 View Other question and answers

Input: user selection

Output: All FAQ with given answers are displayed.

2.5 Manage Statistics

2.5.1 Description and perspective

Admin can view statistics of search results

2.5.2 Functional Requirements

2.5.2.1 View Symptom base records

Input: User selection

Output: all symptom based search result record display in grid view

2.5.2.2 View Disease search records

Input: User selection

Output: all disease search records display in grid view

2.5.2.3 View records in chart form

Input: select chart type

Output: top 5 records from respective categories are displayed as selected chart type

2.6 Manage Others

2.6.1 Description and perspective

Users can read medical related news, see about coronavirus updates, find medical shops

2.6.2 Functional Requirements

2.6.2.1 View Medical related News

Input: User selection

Output: all the news related to medical is displayed

2.6.2.2 View Corona virus Updates

2.6.2.2.1 View All records

Input: user selection

Output: All summary of records is displayed

2.6.2.2.2 View Country wise records

Input: country name

Output: All records of country is displayed

2.6.2.3 Find nearby medical shops

Input: User selection

Output: list is displayed

2.6.2.4 Find medical shops in particular location

Input: Location

Output: list is displayed

3. Non Functional Requirements

3.1 Performance Requirements

User Friendly: The system should be easy to use and easy to understandable.

Scalability: The system should be scalable to handle more users in future.

Data Integrity: The system should maintain and assure data accuracy and consistency over entire lifecycle.

3.2 Security Requirements

User should be authenticated. All the user's data are secure with the system.

3.3 Software Quality Attributes

Flexibility: All the changes that are made by user should be reflected properly in database.

Maintainability: The system can be modified to correct faults, improve performance and prevent unexpected working condition.

4. Software Process Model

We have used the **Incremental Process model** for developing the system.

We have divided the project into modules so that implementation becomes easy. Since we will be implementing in modules testing and debugging will be easy. We believe doing the project in modules will increase our efficiency as we will be concentrating on the functionalities of only one module.

5. Timeline Chart

Week 1: selection of project

Weel2: brief discussion on selected project.

Week 3: functionalities are identified.

Week 4: System Requirement Specification document is prepared.

Week 5: Database design is ready. Registration, login, corona updates module is prepared.(only backend)

Week 6: Remaining GUI part is completed and disease module for doctor is prepared.

Week 7: Design document is prepared (class diagram, use case diagram, activity diagram, sequence diagram). Doctor profile and newsfeed is added.

Week 8: Changes in coronavirus updates done and some test cases are prepared. Basic search for disease based on symptoms is implemented.

Week9: Modification is done for search based on symptoms for more accurate results.

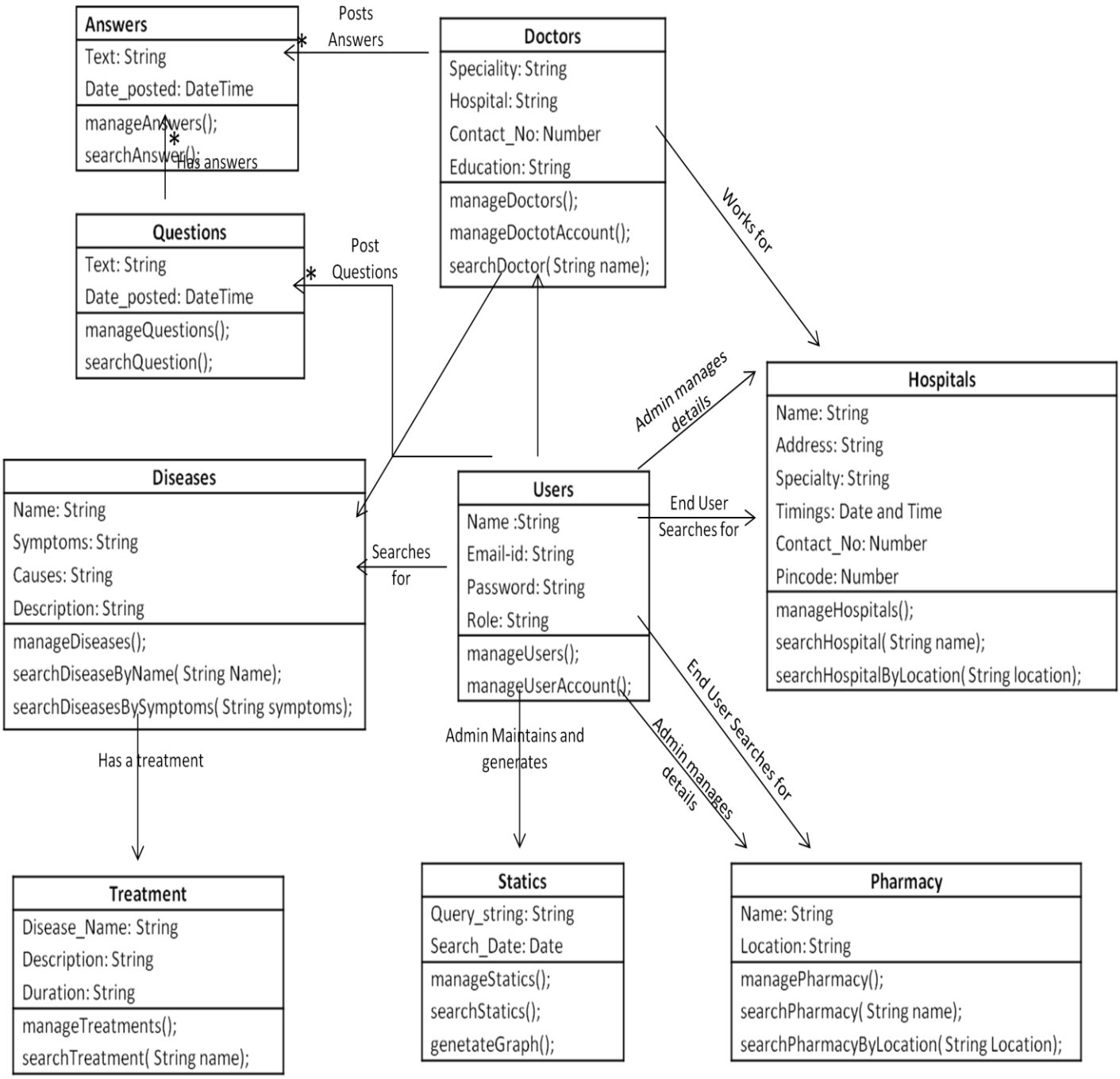
Week10: Hospital and Pharmacy module is done which is managed by admin side. Skin disease are added and viewed from doctor side.

Week11: Disease Treatment module is added and Statistics part on admin side is prepared.

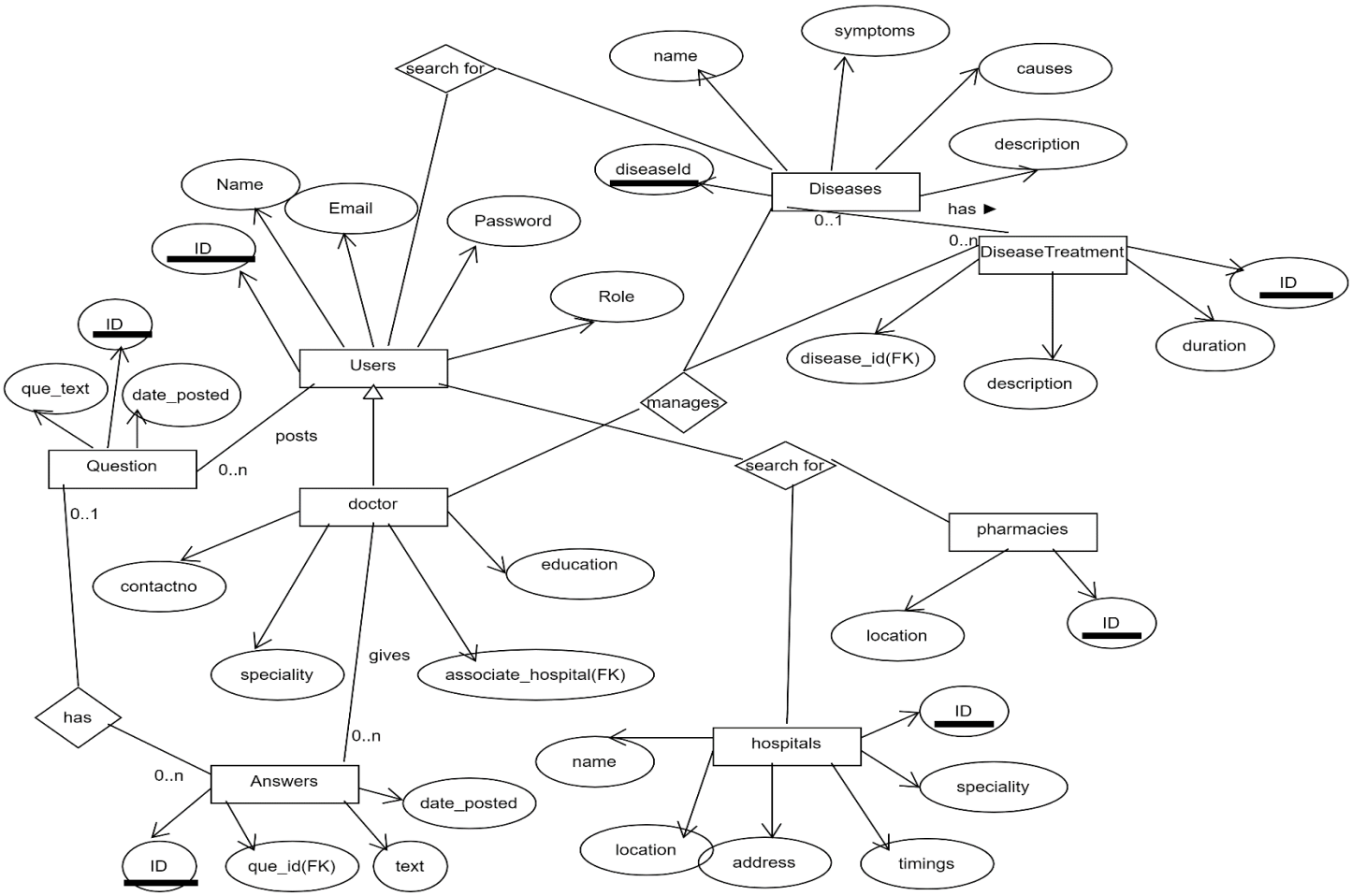
Week12: FAQ module is prepared and some GUI modification for statistics part is done.

Design

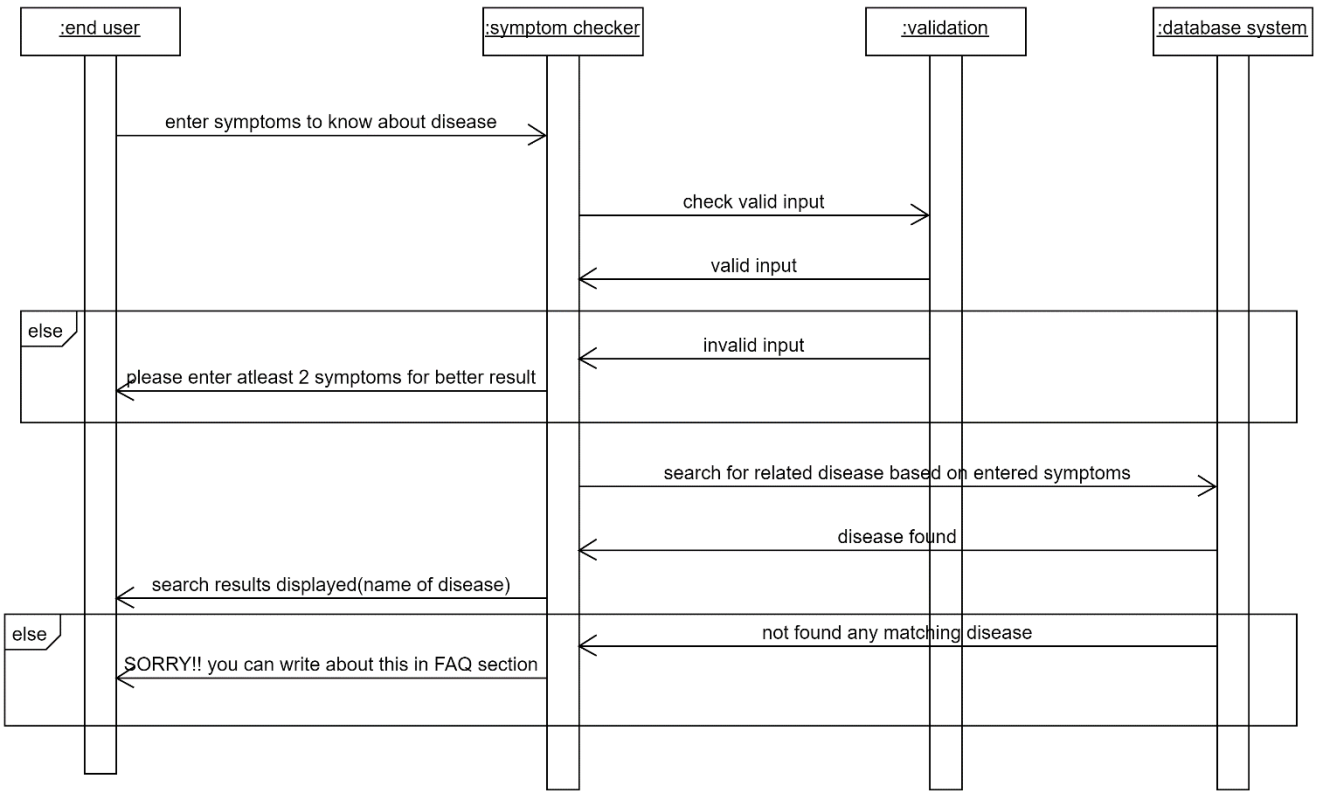
1. Class Diagram



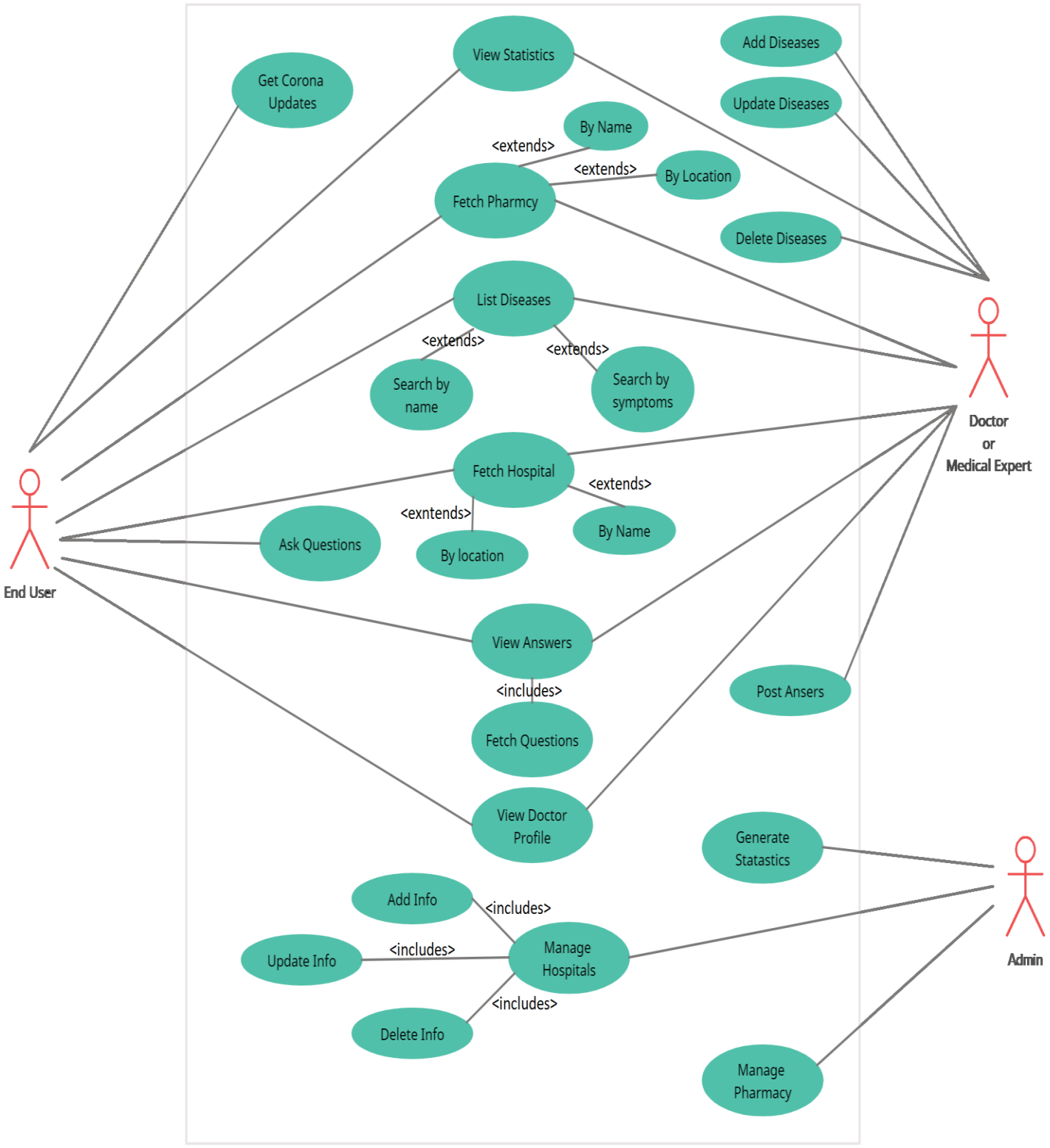
2. ER Diagram



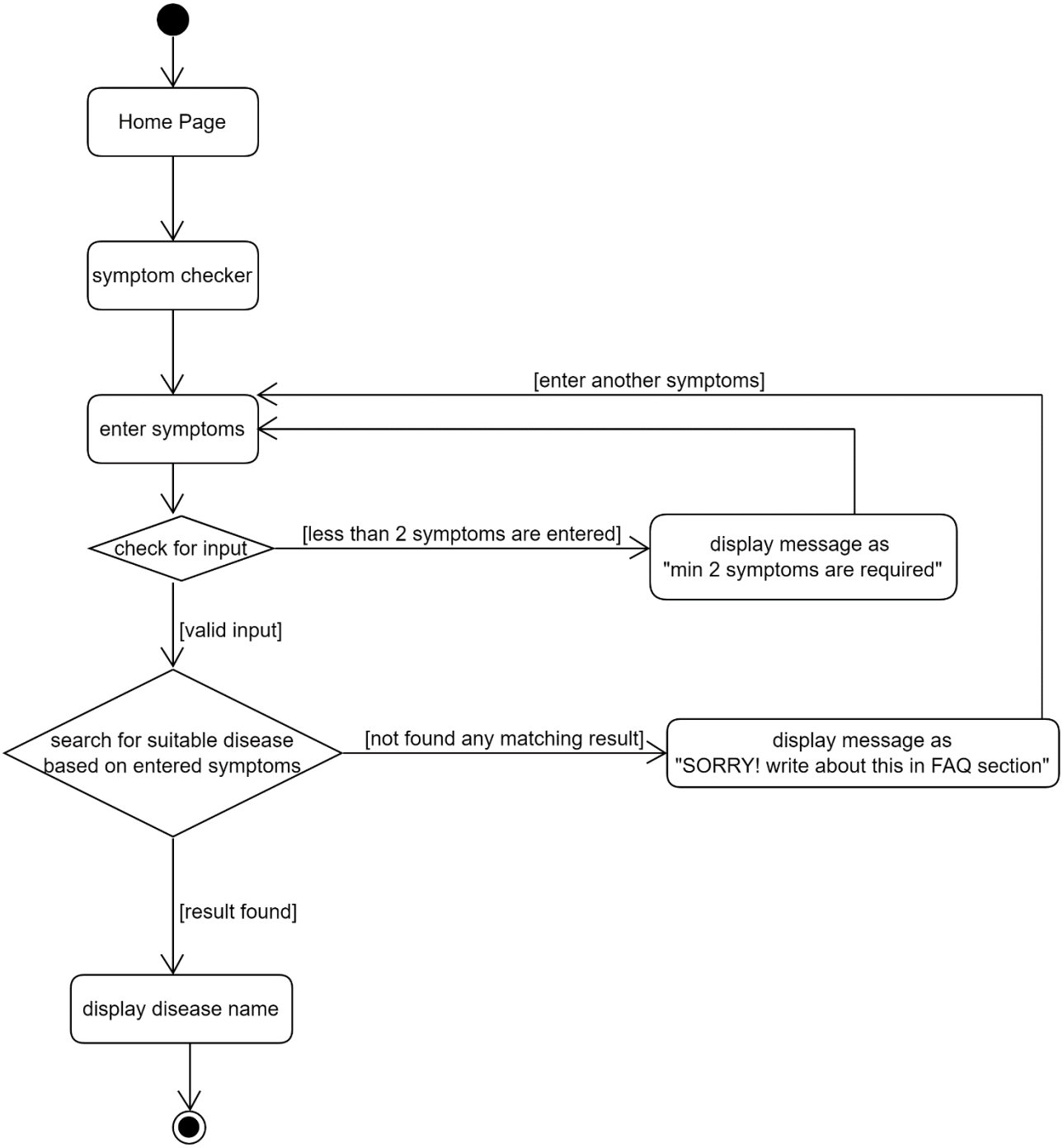
3. Sequence Diagram



4. Use Case Diagram



5. Activity Diagram for Search Disease based on symptoms



6. Database Design

i. User Details

Field	Data Type
User id	Number (Primary Key)
Name	Varchar
Email id	Varchar
Password	Varchar
Role	Varchar

ii. Doctor

Field	Data Type
User_id	User (Foreign Key)
Contact no	Integer
Education	Varchar
Speciality	Varchar
Associate_hospital	Varchar

iii. Disease

Field	Data Type
Disease_id	Number (Primary Key)
Name	Varchar
Symptoms	Varchar
Description	Varchar
Cause	Varchar

iv. Skin Disease

Field	Data Type
Skin_Disease_id	Number (Primary Key)
Name	Varchar
Image1	Image
Image2	Image
Image3	Image

v. Disease Treatment

Field	Data Type
Id	Number (Primary Key)
Disease_id	Number (Foreign Key)
Description	Varchar
Duration	Varchar

vi. Hospital

Field	Data Type
Id	Number (Primary Key)
Name	Varchar
Address	Varchar
Location	Varchar
Speciality	Varchar
Timings	Date-Time
Pincode	Integer

vii. Pharmacy

Field	Data Type
Id	Number (Primary Key)
Name	Varchar
Location	Varchar

viii. Question

Field	Data Type
Que_id	Number (Primary key)
Text	Varchar
Date_posted	DateTime

ix. Answer

Field	Data Type
Answer_id	Number (Primary key)
Que_id	Number(Foreign key)
Text	Varchar
Date_Posted	DateTime

Implementation Details

Modules Implemented

1. Login and Registration Module

- This module handles user's login, logout, role management, registration, change password etc.
- Users first need to provide all required information to register. Authentication is used to make sure that only registered users can access the functionalities.
- Django's auth library is used for authentication.

2. Diseases Module

- This module is implemented to manage diseases, fetch diseases on the basis of symptoms and find its corresponding treatments.
- Doctors can create a record for a new disease, update existing information, delete records and fetch disease based on name.
- Doctors can also upload images for skin related diseases.
- All users can fetch diseases based on symptoms. They need to provide minimum of 2 symptoms and a maximum of 3 symptoms. All diseases in the database are retrieved who have the mentioned symptoms.
- Users can also find treatment for a particular disease based on the name. Once the user enters the name of the disease, the database is searched for the corresponding details.

3. Doctor Module

- This module is implemented to manage doctor information. Doctors can create their profile and associate themselves with a particular hospital. They can update their profiles as well.
- They can also fetch other doctor's profiles.

4. Statistics Module

- This module is used to maintain information regarding the application's statics.
- Admin can view the most frequently searched symptoms and the data is represented in form of pie-charts.
- Whenever a search is made a record is made in the database and using these records the information is generated.
- Charts.js library is used for implementing the pie-charts and other graphs.

5. Hospital Module

- This module is implemented to manage hospital information. Any admin member can create update and delete hospital records.
- Users can fetch hospitals nearby based on their current location. Python's geolocation api is used to retrieve the current location of the user based on their IP. On the basis of the location obtained using the geocoder api the nearby hospitals are fetched.

6. Pharmacy Module

- This module is implemented to manage pharmacy information. Any admin member can create update and delete pharmacy records.
- Users can fetch pharmacies nearby based on their current location. Python's geolocation api is used to retrieve the current location of the user based on their IP. On the basis of the location obtained using the geocoder api the nearby pharmacies are fetched.

7. FAQ Module

- This module is used by users to post question and the doctors post answers to these questions.
- Users can see all the questions added in the last 3 days. However they can fetch other questions from the search bar.

8. News Module

- Google's news api is used to fetch all the recent medical related information.
- The api is used in a manner such that only medicine and health related news is displayed.

9. Corona Virus Updates Module

- Django's covid module is used to implement this module. Using this module data is retrieved and displayed.
- The data can be fetched for a particular country. Chart.js is used to represent the data in graphical forms.
- Also data for all countries can be fetched.

Function Prototypes

1. Fetch Diseases based on symptoms

```
def search(request):
  c={}
  c.update(csrf(request))
  sym1=(int)(request.POST['sym1'])
  sym2=(int)(request.POST['sym2'])
  if(sym1==0 or sym2==0):
    return render(request,'search.html',{'c':c,'found':False,'l1':symp,'errmsg':'enter at least 2 symptoms '})
  s1=SearchSymptomRecord.objects.filter(symptom=symp[sym1-1])
  s2=SearchSymptomRecord.objects.filter(symptom=symp[sym2-1])
  if(not s1):
    sym=SearchSymptomRecord(symptom=symp[sym1-1],searchcount=1)
    sym.save()
  else:
    sym=SearchSymptomRecord.objects.get(symptom=symp[sym1-1])
    sym.searchcount=sym.searchcount+1
    sym.save()
```

```
if(not s2):
   sym=SearchSymptomRecord(symptom=symp[sym2-1],searchcount=1)
   sym.save()
else:
   sym=SearchSymptomRecord.objects.get(symptom=symp[sym2-1])
   sym.searchcount=sym.searchcount+1
   sym.save()
ds1=Disease.objects.filter(Q(Symptoms__icontains=symp[sym1-1]))
ds2=Disease.objects.filter(Q(Symptoms__icontains=symp[sym2-1]))
sym3=(int)(request.POST['sym3'])
if(sym3==0):
   if(ds1 or ds2):
      q1=set(ds1)
      q2=set(ds2)
      q=set(ds1).intersection(set(ds2))
      if(q):
         for d in q:
            d1=SearchDiseaseResult.objects.filter(diseasename=d)
            if(not d1):
               d1=SearchDiseaseResult(diseasename=d,searchcount=1)
               d1.save()
            else:
               d1=SearchDiseaseResult.objects.get(diseasename=d)
               d1.searchcount=d1.searchcount+1
               d1.save()
         return render(request,'search.html',{'q':q,'found':True,'c':c,'l1':symp})
      else:
         for d in q1:
            d1=SearchDiseaseResult.objects.filter(diseasename=d)
            if(not d1):
               d1=SearchDiseaseResult(diseasename=d,searchcount=1)
               d1.save()
            else:
               d1=SearchDiseaseResult.objects.get(diseasename=d)
               d1.searchcount=d1.searchcount+1
               d1.save()
         for d in q2:
            d1=SearchDiseaseResult.objects.filter(diseasename=d)
            if(not d1):
               d1=SearchDiseaseResult(diseasename=d,searchcount=1)
               d1.save()
            else:
```

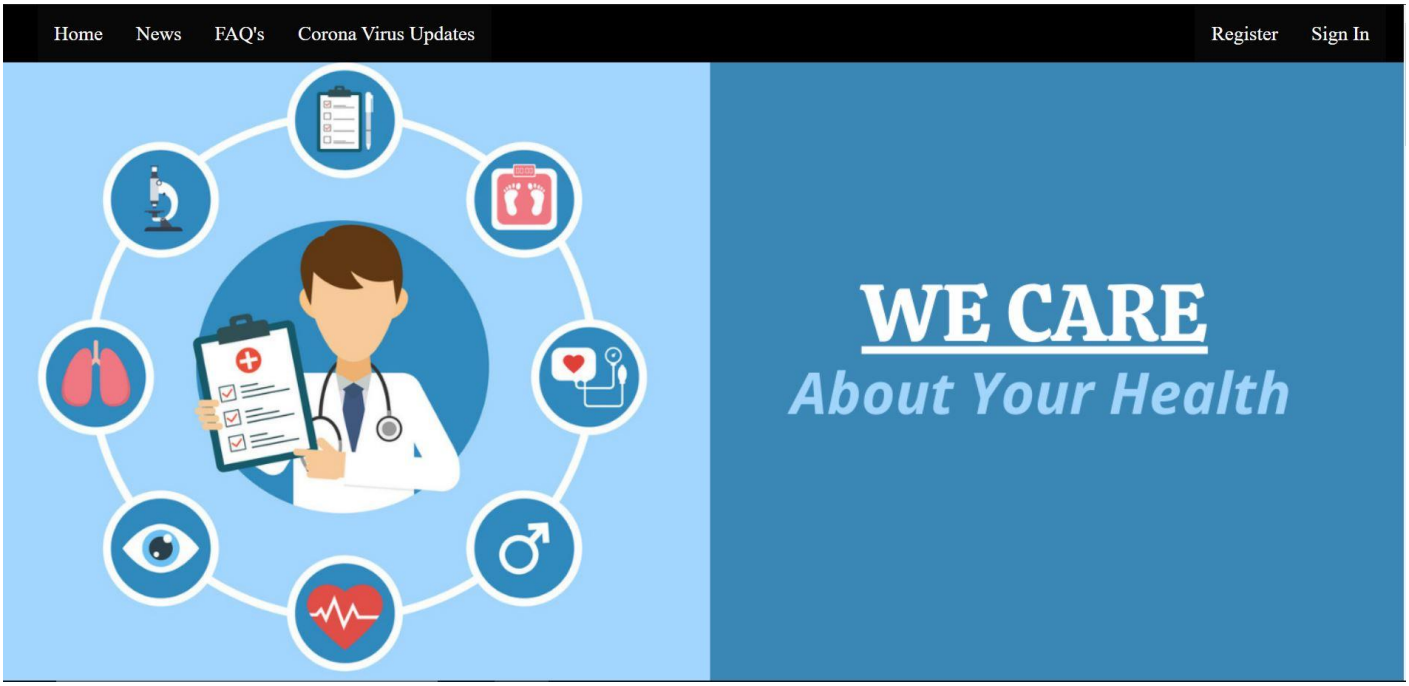
```
                d1=SearchDiseaseResult.objects.get(diseasename=d)
                d1.searchcount=d1.searchcount+1
                d1.save()
            return render(request,'search.html',{'q1':q1,'found':True,'c':c,'q2':q2,'l1':symp,'sug':'provide more
symptoms to get perfect results if possible'})
    else:
        return render(request,'search.html',{'msg':'Sorry!!!not found any matching results..kindly request you
to provide this details in FAQ. our team will give you satisfiable answer there..','found':False,'c':c,'l1':symp})
    else:
      ds3=Disease.objects.filter(Q(Symptoms__icontains=symp[sym3-1]))
      if(ds3 or ds2 or ds1):
        q3=set(ds3)
        q1=set(ds1)
        q2=set(ds2)
        q=q1.intersection(q2.intersection(q3))
        if(q):
          return render(request,'search.html',{'q':q,'found':True,'c':c,'l1':symp})
        q12=q1.intersection(q2)
        q13=q1.intersection(q3)
        q23=q2.intersection(q3)
        if(q12 or q13 or q23):
         return render(request,'search.html',{'q12':q12,'q13':q13,'q23':q23,'found':True,'c':c,'l1':symp})
        else:
          return render(request,'search.html',{'q1':q1,'found':True,'c':c,'q2':q2,'q3':q3,'l1':symp,'sug':'provide
matching symptoms to get suitable results if possible'})
      else:
        if(ds1 or ds2):
          q1=set(ds1)
          q2=set(ds2)
          q=set(ds1).intersection(set(ds2))

          if(q):
            return render(request,'search.html',{'q':q,'found':True,'c':c,'l1':symp})
          else:
            return render(request,'search.html',{'q1':q1,'found':True,'c':c,'q2':q2,'l1':symp,'sug':'provide more
symptoms to get perfect results if possible'})
        else:
          return render(request,'search.html',{'msg':'Sorry!!!not found ', 'found' :False, 'c':c,'l1':symp})
    return render(request,'search.html')
```

2. Search nearby Hospital or Pharmacy

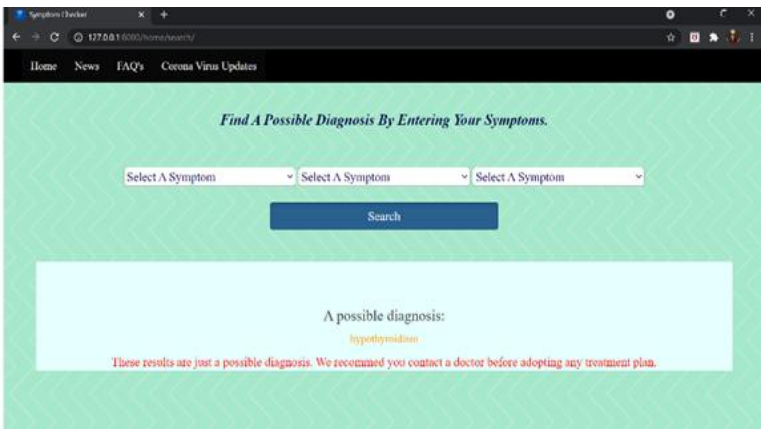
```
def pharmacysearchnearest(request):
    object_list=Pharmacy.objects.all()
    sorted_list=[]
    for pharm in object_list:
        g=geocoder.ip('me')
        geolocator=Nominatim(user_agent='WeCare')
        location=geolocator.geocode(pharm.location)
        print(pharm.location)
        location2=(location.latitude,location.longitude)
        location3=(g.lat,g.lng)
        print(location.address)
        print(g.address)
        print((location.latitude,location.longitude))
        print((g.lat,g.lng))
        pharm.distance=geodesic(location2,location3).km

    for pharm in object_list:
        info={
            "name":pharm.name,
            "location":pharm.location,
            "distance":pharm.distance
        }
        sorted_list.append(info)
    s_list=sorted(sorted_list,key=lambda i:i['distance'])
    print(s_list)
    return render(request,'nearestpharmacy.html',{'objectlist':s_list,'found':True})
```

Screenshots

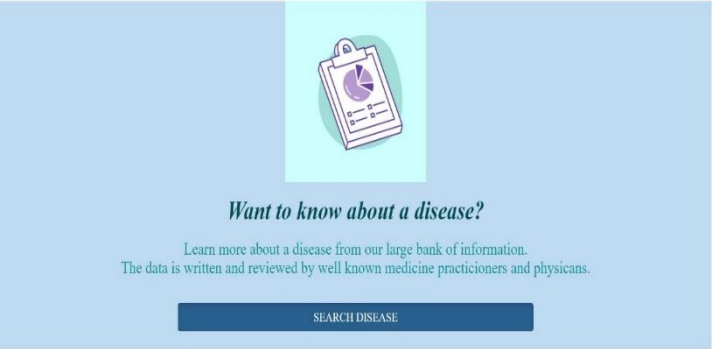
1. Home Page

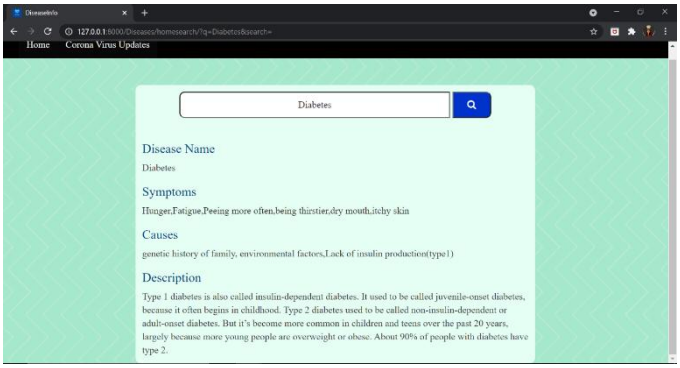


1.1 Symptom checker

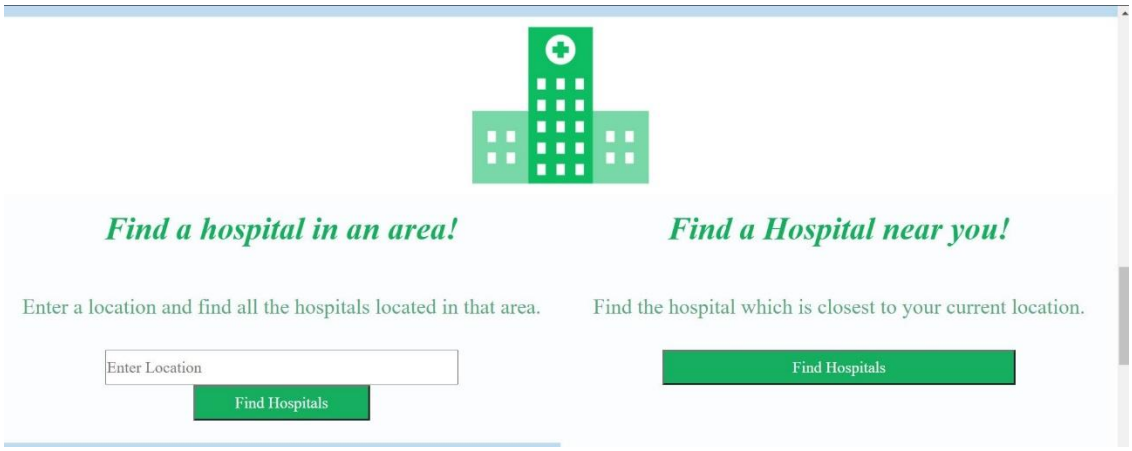


1.2 Search for disease by name

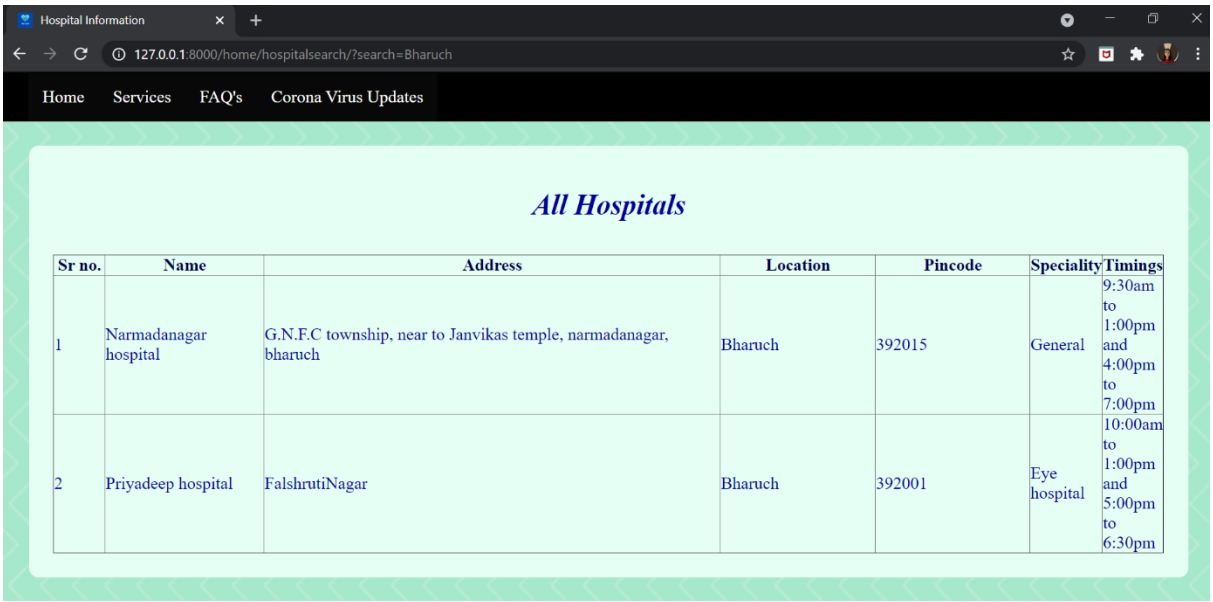




1.3 Search nearby hospital or hospital in particular location



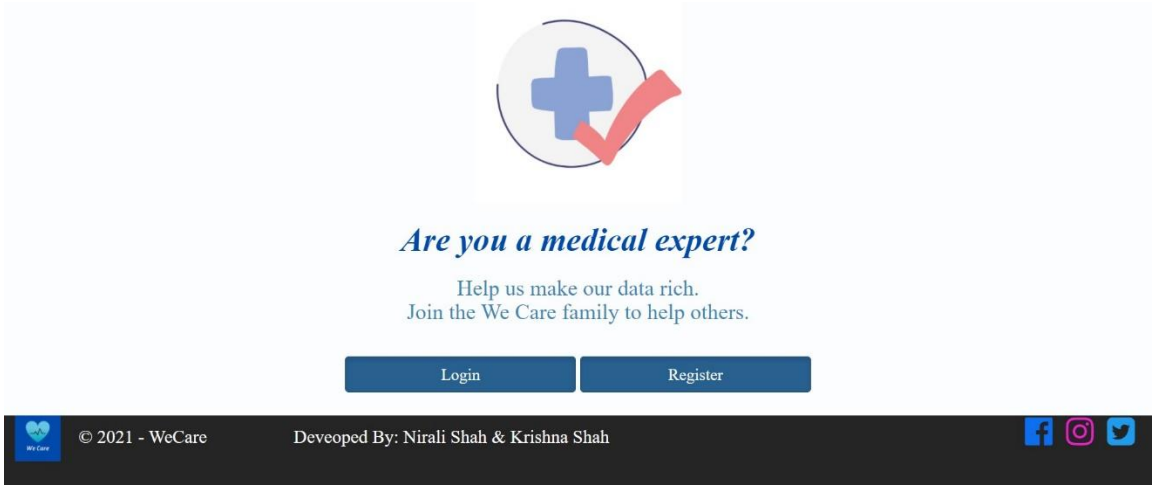
- **Search results for hospital in particular location**



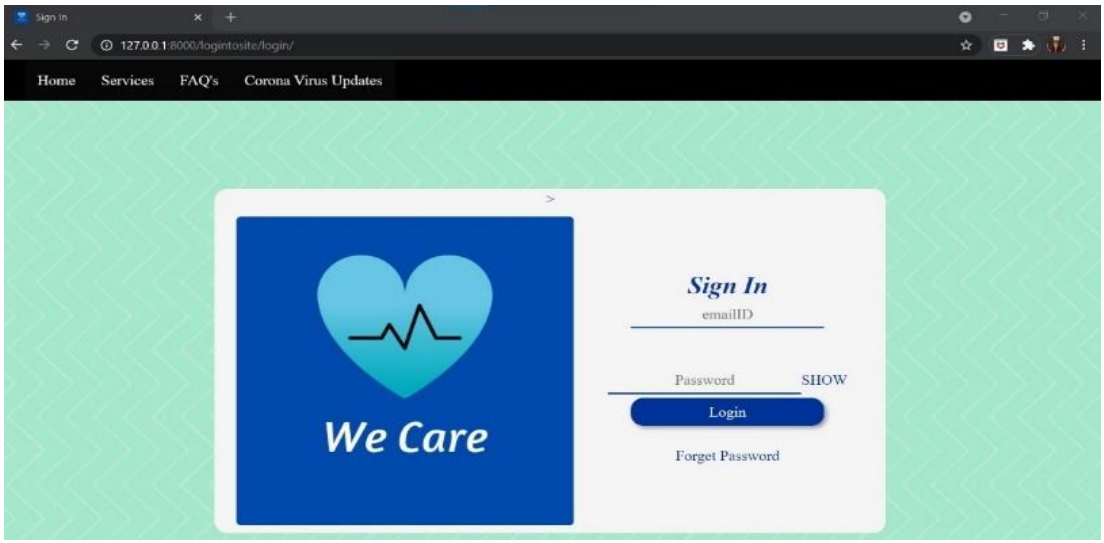
Sr no.	Name	Address	Location	Pincode	Speciality	Timings
1	Narmadanagar hospital	G.N.F.C township, near to Janvikas temple, narmadanagar, bharuch	Bharuch	392015	General	9:30am to 1:00pm and 4:00pm to 7:00pm
2	Priyadeep hospital	FalshrutiNagar	Bharuch	392001	Eye hospital	10:00am to 1:00pm and 5:00pm to 6:30pm

1.4Search nearby pharmacies or pharmacies in particular location

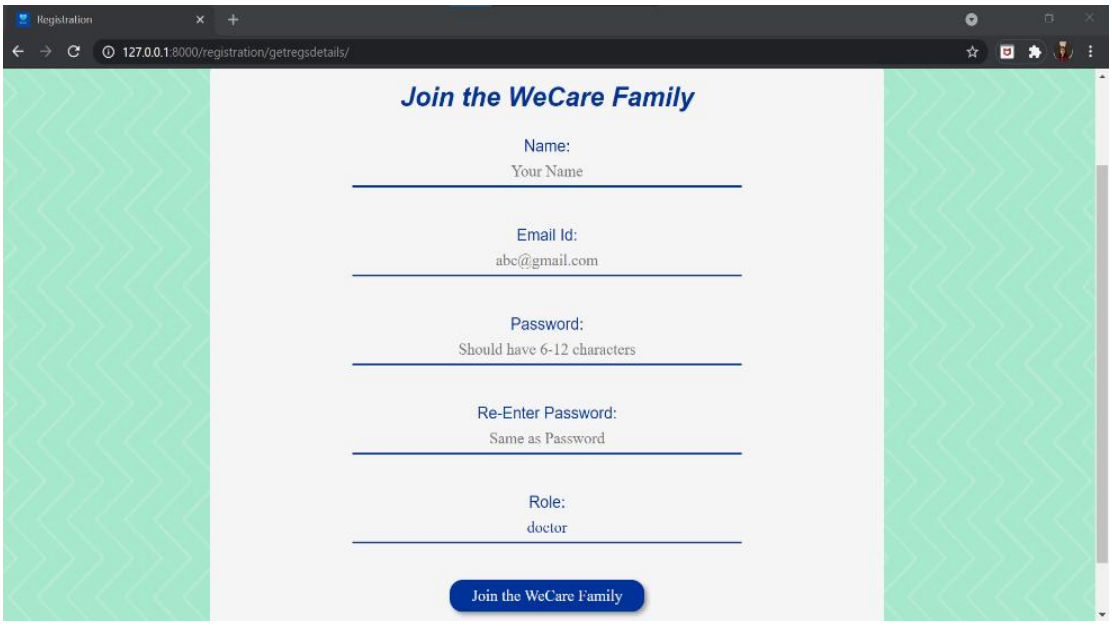
1.5 Register yourself or login



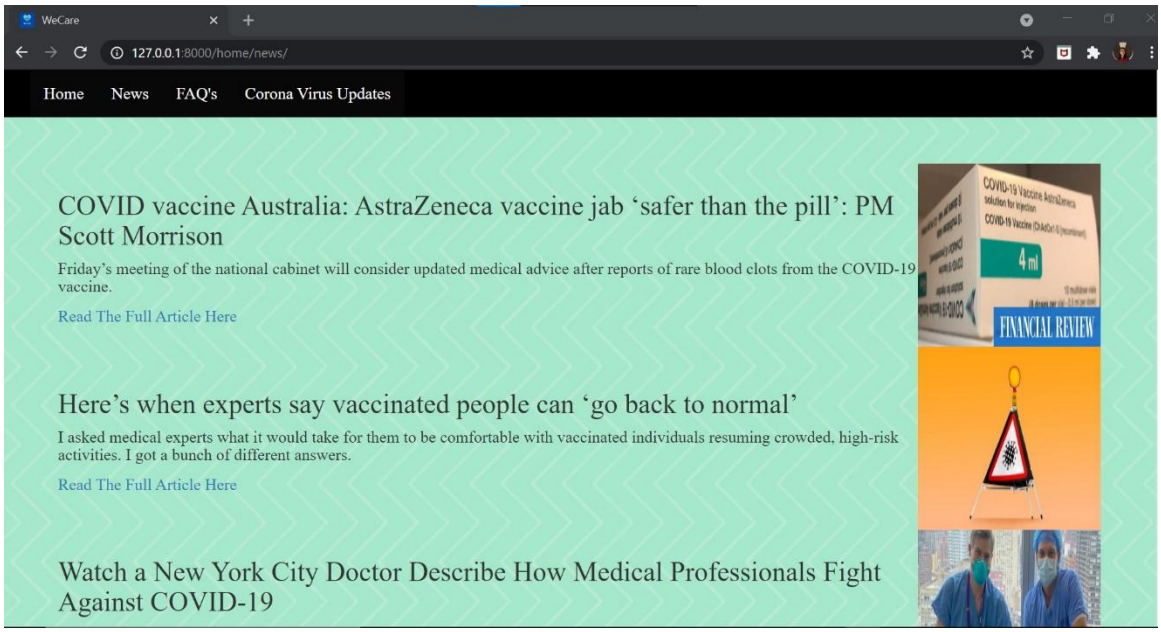
- **Login**



- **Registration**

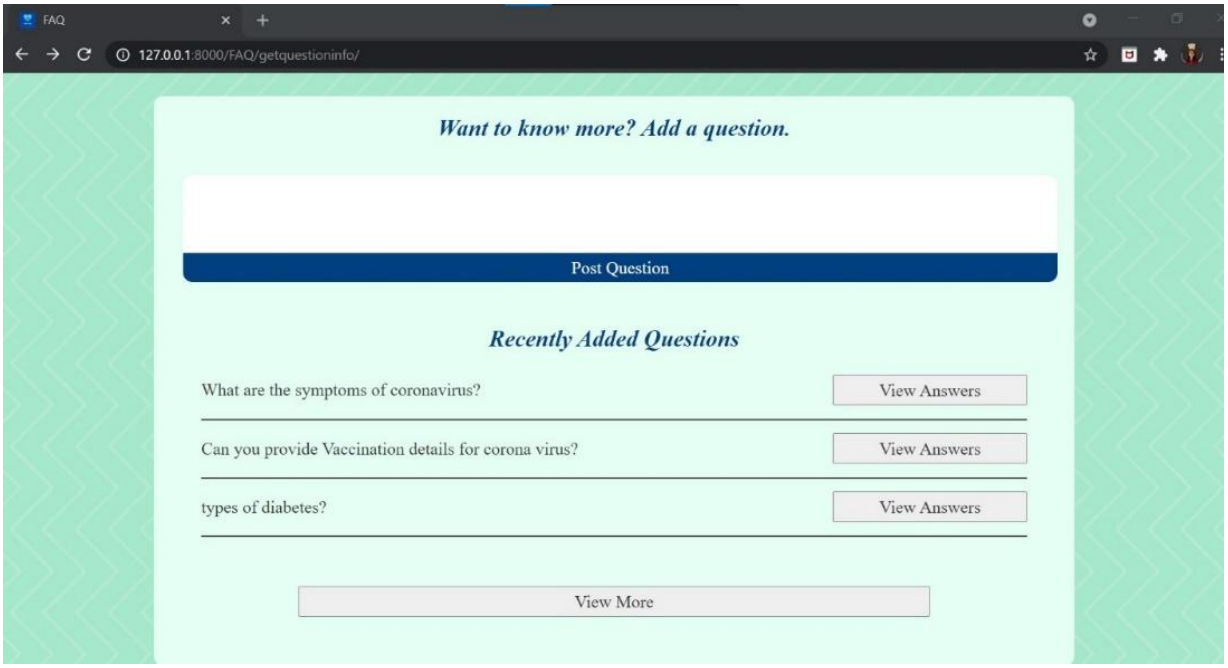


2. News

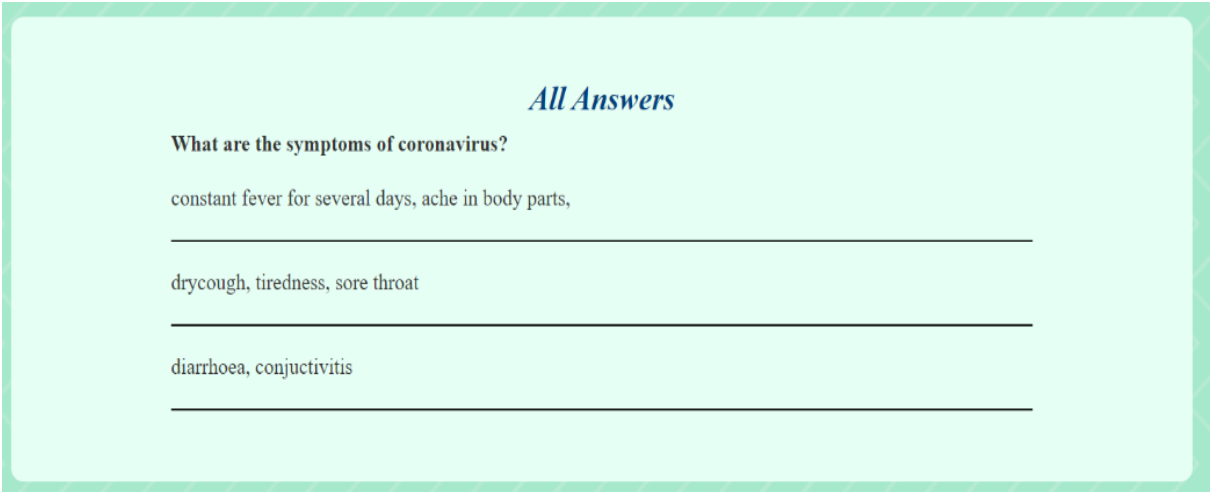


3. FAQ

3.1User Adds Question

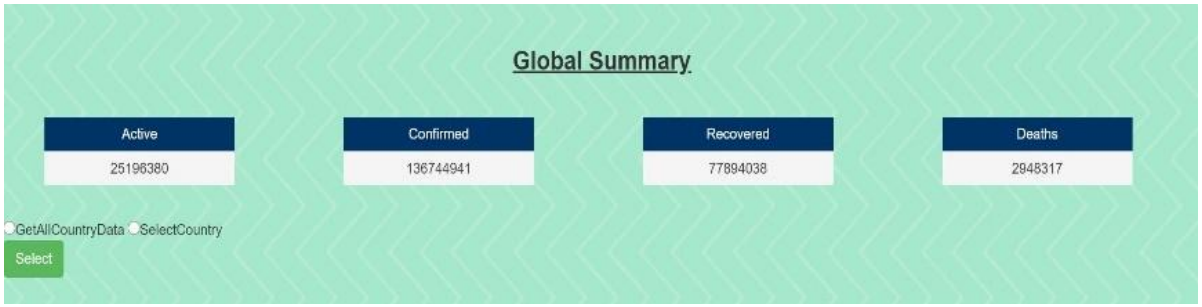


3.2 View all answers of a particular question

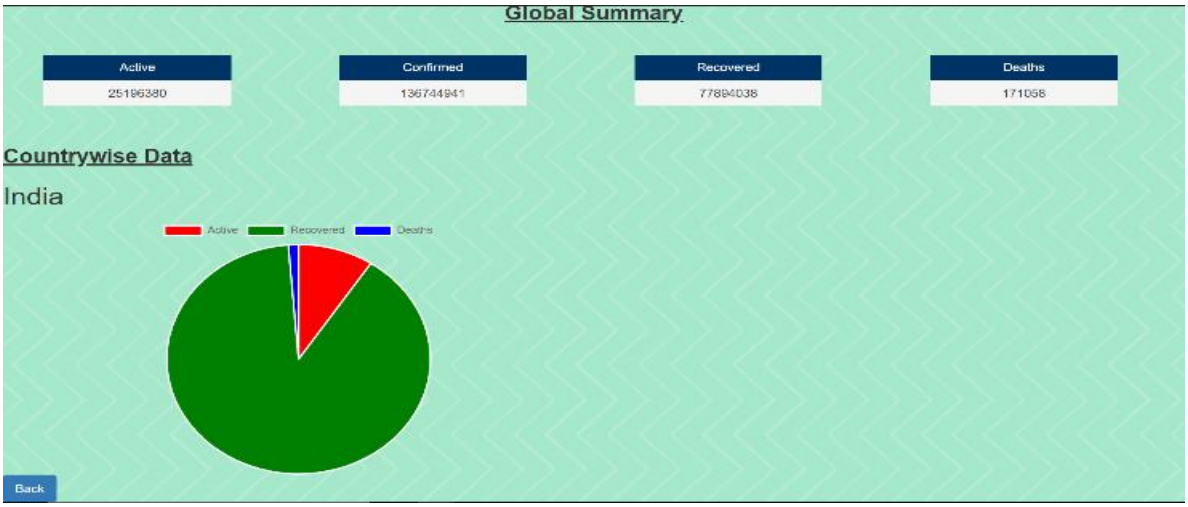


4. Coronavirus Updates

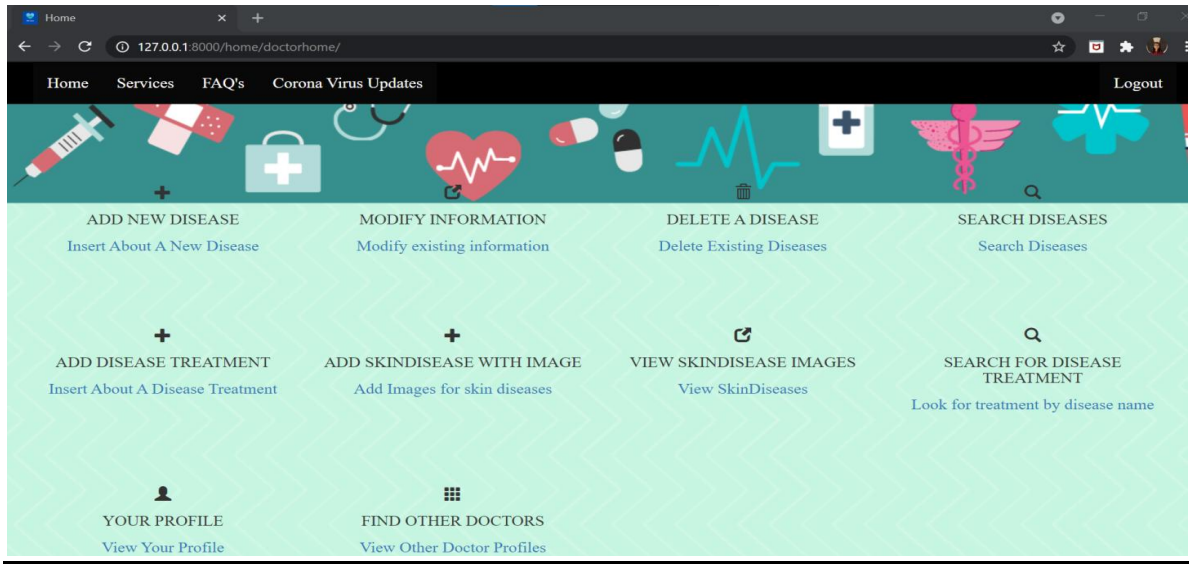
- **Global summary**



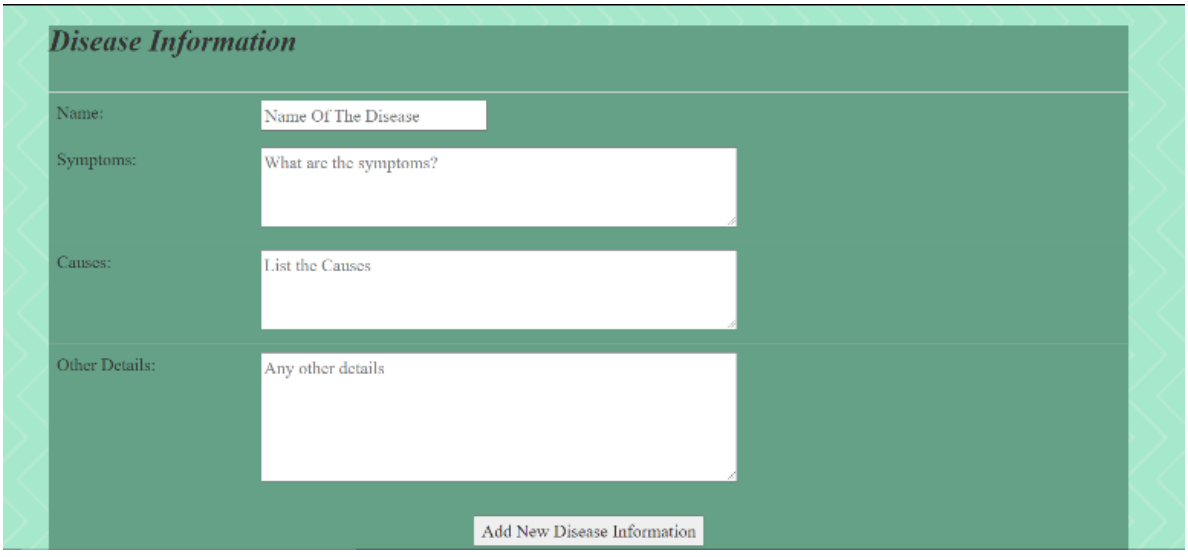
- **After selecting country name(country wise data)**



5. Doctor's Homepage



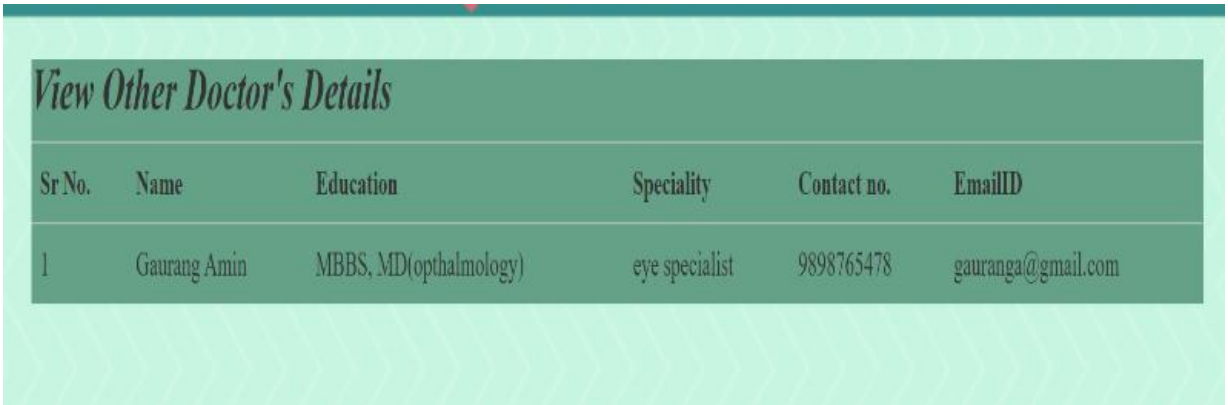
6. Add new Disease



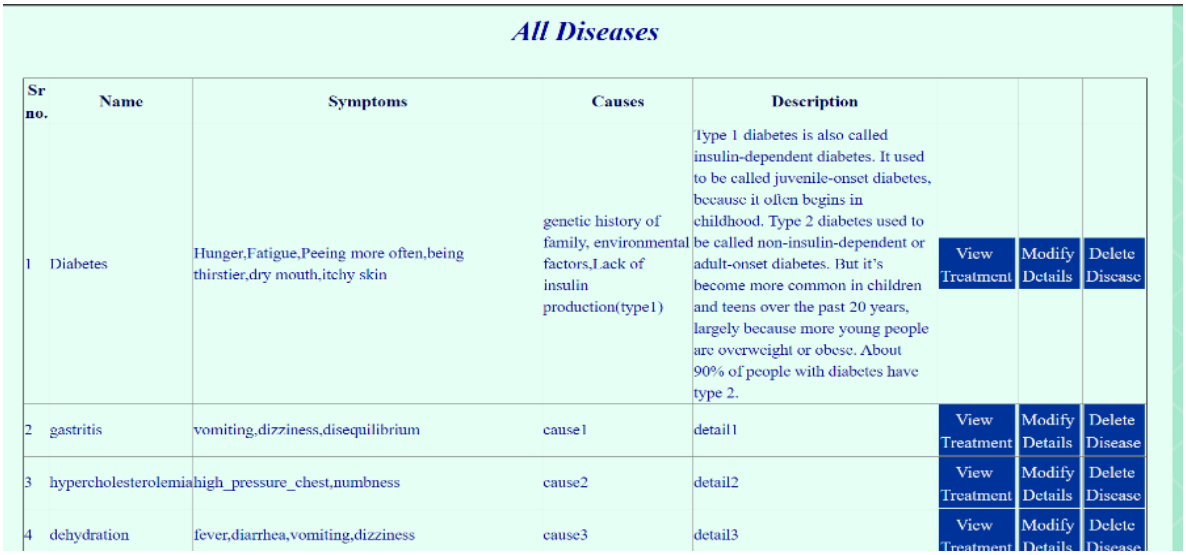
7. Add disease treatment

8. View or Update Profile

9. View other doctor's details



10.View Disease information



All Diseases

Sr no.	Name	Symptoms	Causes	Description			
1	Diabetes	Hunger,Fatigue,Peeing more often,being thirstier,dry mouth,itchy skin	genetic history of family, environmental factors,Lack of insulin production(type1)	Type 1 diabetes is also called insulin-dependent diabetes. It used to be called juvenile-onset diabetes, because it often begins in childhood. Type 2 diabetes used to be called non-insulin-dependent or adult-onset diabetes. But it's become more common in children and teens over the past 20 years, largely because more young people are overweight or obese. About 90% of people with diabetes have type 2.	View Treatment	Modify Details	Delete Disease
2	gastritis	vomiting,dizziness,disequilibrium	cause1	detail1	View Treatment	Modify Details	Delete Disease
3	hypercholesterolemia	high_pressure_chest,numbness	cause2	detail2	View Treatment	Modify Details	Delete Disease
4	dehydration	fever,diarrhea,vomiting,dizziness	cause3	detail3	View Treatment	Modify Details	Delete Disease

11.Modify Disease Information

Disease Information

Name: Diabetes

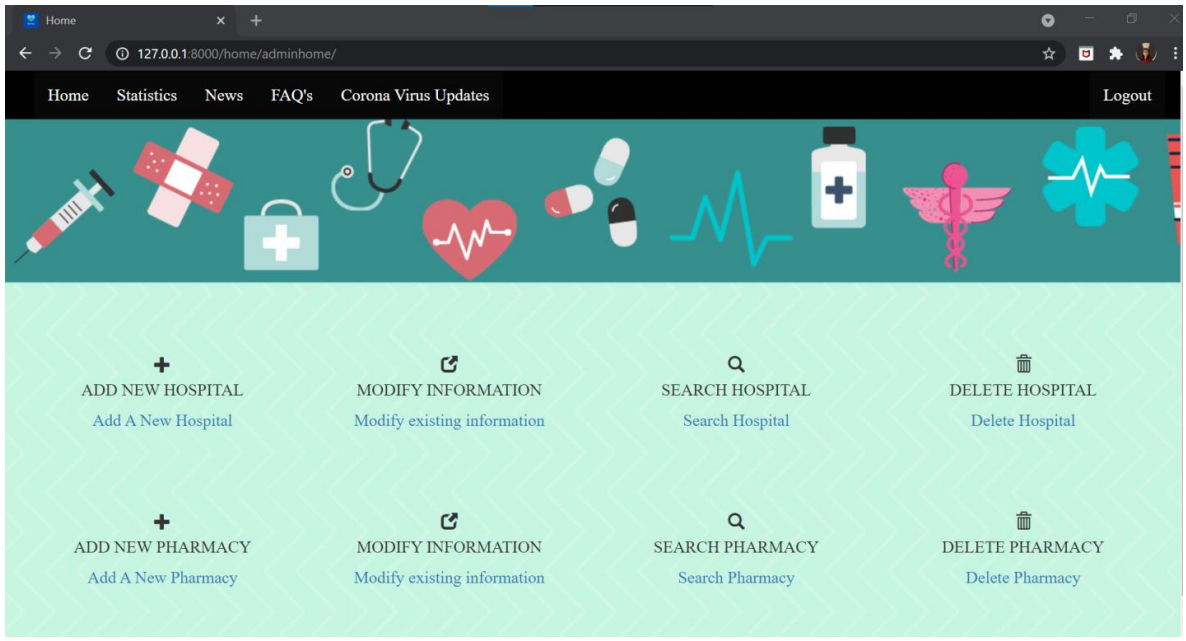
Symptoms: Hunger,Fatigue,Peeing more often,being thirstier,dry mouth

Causes: genetic history of family, environmental factors,Lack of insı

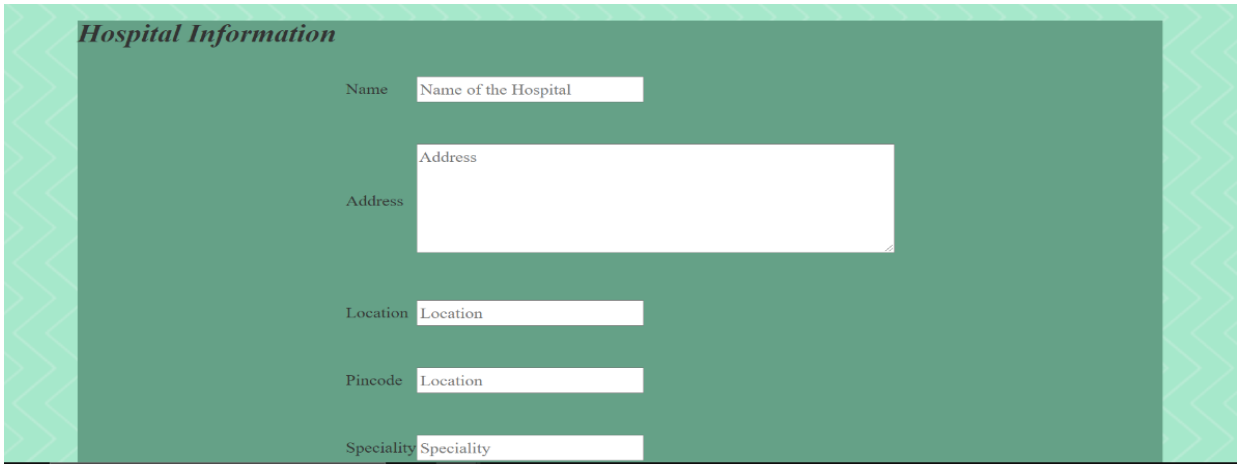
Description: Type 1 diabetes is also called insulin-dependent diabetes. It

Update Details

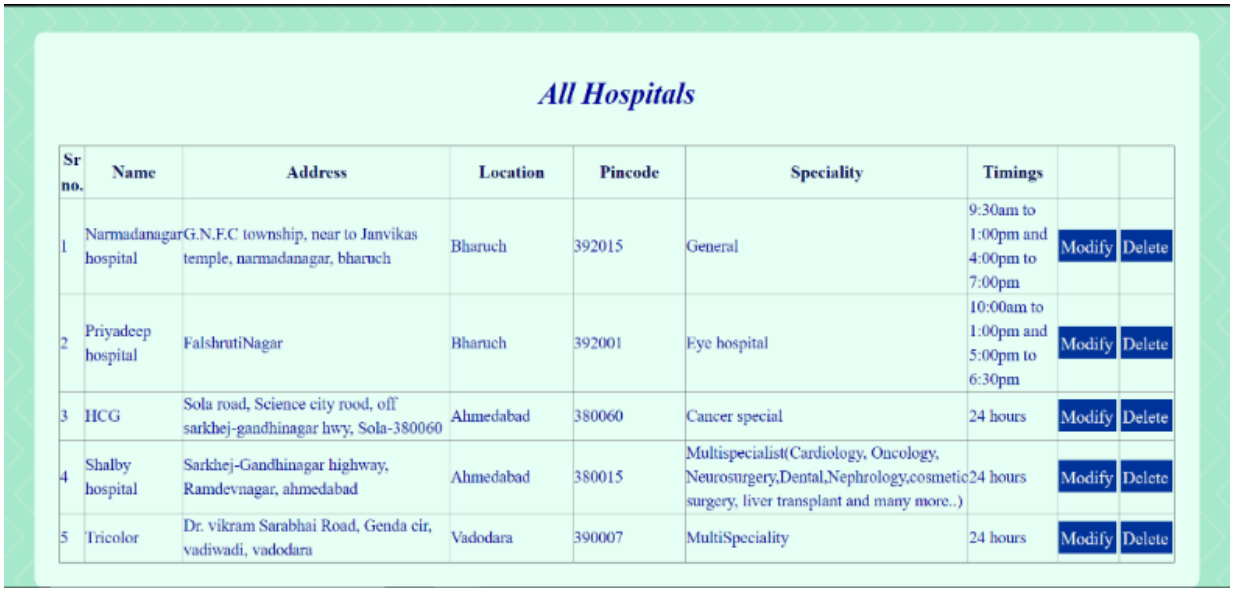
12.Admin Homepage



13. Add hospital

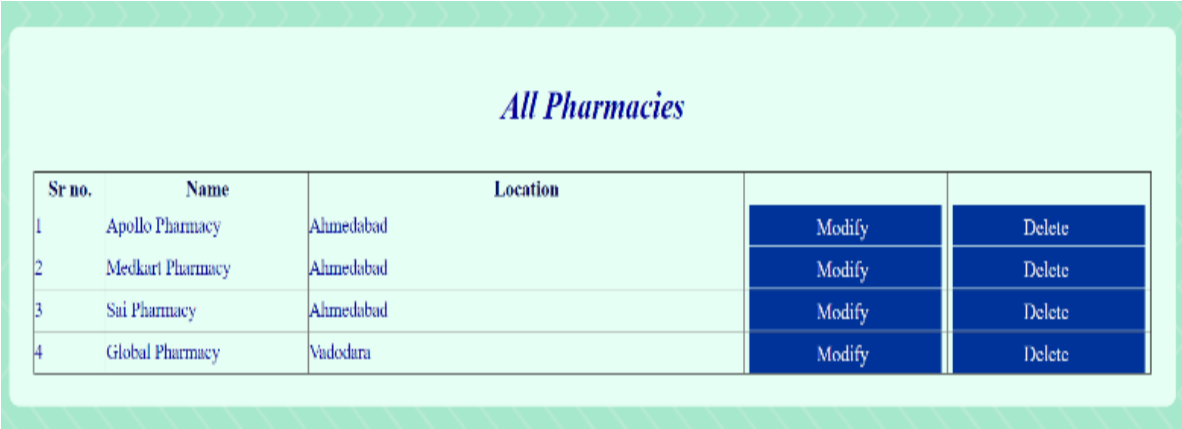


14.View All hospitals



Sr no.	Name	Address	Location	Pincode	Speciality	Timings		
1	Narmadanagar hospital	G.N.F.C township, near to Janvikas temple, narmadanagar, bharuch	Bharuch	392015	General	9:30am to 1:00pm and 4:00pm to 7:00pm	Modify	Delete
2	Priyadeep hospital	FalshrutiNagar	Bharuch	392001	Eye hospital	10:00am to 1:00pm and 5:00pm to 6:30pm	Modify	Delete
3	HCG	Sola road, Science city rood, off sarkhej-gandhinagar hwy, Sola-380060	Ahmedabad	380060	Cancer special	24 hours	Modify	Delete
4	Shalby hospital	Sarkhej-Gandhinagar highway, Ramdevnagar, ahmedabad	Ahmedabad	380015	Multispecialist(Cardiology, Oncology, Neurosurgery,Dental,Nephrology,cosmetic surgery, liver transplant and many more..)	24 hours	Modify	Delete
5	Tricolor	Dr. vikram Sarabhai Road, Genda cir, vadiwadi, vadodara	Vadodara	390007	MultiSpeciality	24 hours	Modify	Delete

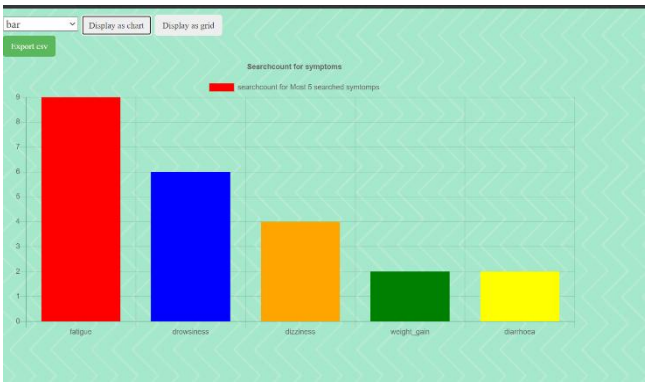
15. View All Pharmacies

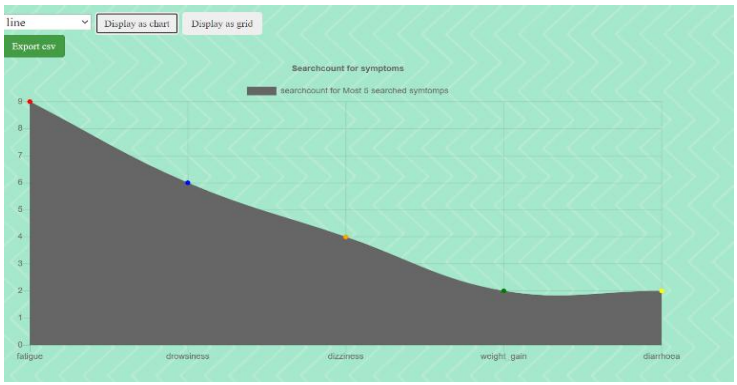


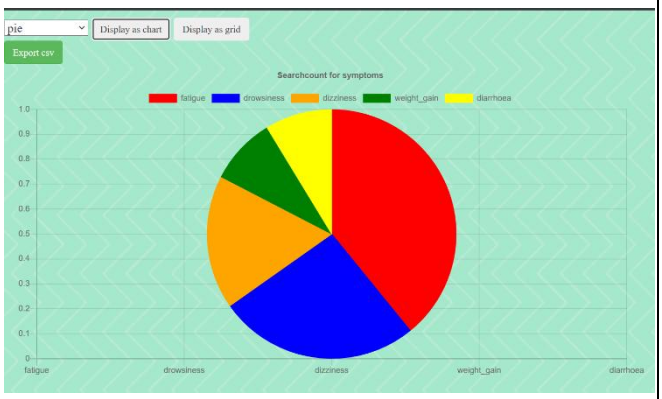
16.Statistics

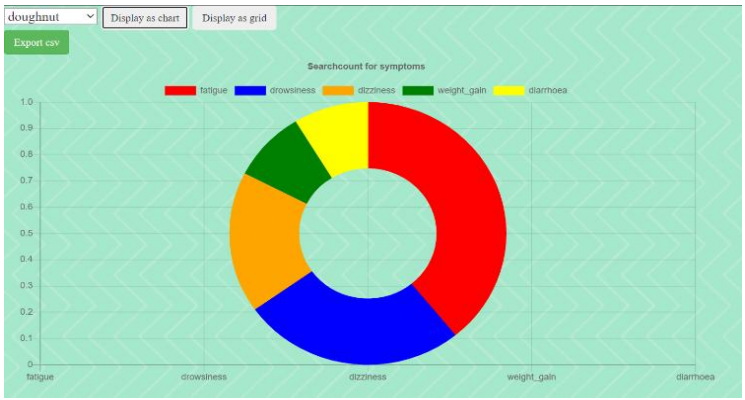
- **Grid view**

- **Graph View**


bar
Display as chart
Display as grid
Export csv
Searchcount for symptoms
searchcount for Most 5 searched symtomps
fatigue
drowsiness
dizziness
weight_gain
diarrhoea


line
Display as chart
Display as grid
Export csv
Searchcount for symptoms
searchcount for Most 5 searched symtomps
fatigue
drowsiness
dizziness
weight_gain
diarrhoea


pie
Display as chart
Display as grid
Export csv
Searchcount for symptoms
fatigue
drowsiness
dizziness
weight_gain
diarrhoea


doughnut
Display as chart
Display as grid
Export csv
Searchcount for symptoms
fatigue
drowsiness
dizziness
weight_gain
diarrhoea

Testing

- Unit testing of each module was done after successfully completing the module. Each module was tested individually before integrating them with the whole system. Integration testing was done in order to check if modules are working properly together. Black box testing was done to check if the applications work properly.

Black box testing of Major functions of the system

1. Login

Sr No.	Test Data/ Input	Output
1.	Username and password are valid	Redirected to Admin's home page if Admin Else to Doctor's home page
2.	Valid Username but invalid password	Invalid Password Message Displayed All Fields are reset
3.	Invalid Username but valid password	User doesnot exist message All fields are reset
4.	Invalid Username and Password	User doesnot exist message All fields are reset

2. Registration

Sr. No	Test Data/Input	Output
1.	All fields are filled. Password matches the re-password. Password is of 6-12 characters. User doesn't already exist	Doctor is successful registered. Redirection to Login page.
2.	All fields are filled. Password is of 6-12 characters. Password and re-password doesn't match. User doesn't already exist.	Both Password should match message is displayed. All fields are reset.
3.	All fields are filled. Password is not of 6-12 characters. Password and re-password do match. User doesn't already exist.	Password should be of 6-12 characters message displayed. All fields are reset.
4.	All fields are filled. Password is of 6-12 characters but should be according to regex: [a-zA-Z0-9]. Password and re-password match. User doesn't already exist.	Password Credentials don't match message displayed. All fields are reset.
5.	All fields are filled. Password matches the re-password. Password is of 6-12 characters. User already exists.	User already exists message displayed. All fields are reset.
6.	All fields are not filled.	Please fill this field.

3. **Corona Virus Updates**

Sr. No	Test Data/ Input	Output
1.	Two Countries to compare	Comparison of cases and recovery rate of two countries.
2.	Country Name	Detailed information cases and recovery rate.

4. **Search Disease by name**

Sr. No	Test Data/Input	Output
1.	Disease Exists	Information about disease is displayed.
2.	Disease Doesn't Exist	Disease not found message displayed.
3.	Search field empty	"Enter appropriate search value "message displayed.

5. **Add New Disease Information**

Sr. No	Test Data/ Input	Output
1.	All fields are filled. Disease doesn't already exist.	Record is created. Redirected to home page.
2.	All fields are not filled. Disease doesn't already exist.	All fields are required message displayed.
3.	All fields are filled. Disease already exists.	"Information for this disease already exist" message displayed.

6. **Update Disease Information**

Sr. No	Test Data/ Input	Output
1.	Disease exists. Modified data entered.	Database is updated. Redirected to all diseases list.
2.	Disease doesn't exist.	All diseases list is displayed.

7. **Delete Disease**

Sr. No	Test Data/ Input	Output
1.	Disease exists.	Disease is deleted. Redirected to home page.
2.	Disease doesn't exist.	Disease doesn't exist message displayed.

8. **Create Doctor Profile**

Sr No.	Test Data/ Input	Output
1.	All fields are filled. Name used for registration matches with the name in the profile. Profile photo uploaded.	Profile is created. Redirection to doctor's homepage.
2.	All fields are filled. Name used for registration doesn't match with the name in the profile.	No such user exist message displayed. All fields are reset.

	Profile photo uploaded.	
3.	All fields are filled. Name used for registration matches with the name in the profile. Profile photo not uploaded.	Profile Photo not uploaded message displayed. Profile is created. Redirection to doctor's homepage.
4.	All fields are not filled.	Fill all fields messaged displayed.

9. Update Profile

Sr. No	Test Data/ Input	Output
1.	Profile already created. Modified data is entered.	Changes made to the profile. Redirected to view profile.
2.	Profile is not created.	Redirected to create profile.

10. Delete Profile

Sr. No	Test Data/ Input	Output
1.	Profile exists.	Profile is deleted. Redirected to home page.
2.	Profile doesn't exist.	Profile doesn't exist message displayed.

11. Search Doctor Profile

Sr. No	Test Data/ Input	Output
1.	Profile exists.	Profile is displayed.
2.	Profile doesn't exist.	Profile doesn't exist. Search bar is cleared.

12. New Hospital Record

Sr. No	Test Data/ Input	Output
1.	All fields are filled. Record doesn't already exist.	New record is made. Redirected to home page.
2.	All fields are not filled. Record doesn't already exist.	Please fill the field message displayed.
3.	All fields are filled. Record already exists.	Record already exists message displayed. All fields are reset.

13. Update Hospital Record

Sr. No	Test Data/ Input	Output
1.	Hospital Record exists. Modified data is entered.	Changes made to the record. Redirected to view hospitals.
2.	Hospital Record doesn't exist.	Redirected to create record.

14. Delete Record

Sr. No	Test Data/ Input	Output
1.	Record exists.	Record is deleted. Redirected to home page.
2.	Record doesn't exist.	Hospital Record doesn't exist message displayed.

15. Search Hospital

Sr. No	Test Data/ Input	Output
	For Admin and doctor	
1.	Hospital name [Hospital Record Exists]	Information about the hospital is displayed.
2.	Hospital name [Hospital Record Doesn't Exist]	Record not found message displayed.
	For end users	
3.	Location [If hospital records for that location are found]	List of Hospitals in specified location is displayed
4.	Location[Records for that location does not exist]	Empty list displayed
5.	User selection for nearby hospital	List of hospital in nearby location of your machine's ip address. [If found]
6.	Search field empty	"Enter appropriate search value "message displayed.

16. Pharmacy Search

Sr. No	Test Data/ Input	Output
1.	Location	List of medical shops
2.	User selection for nearby location	List of medical shops in nearby location of your machine's ip address. [If found]
3.	Search field empty	"Enter appropriate search value "message displayed.

17. Search disease based on symptoms

Sr. No.	Test Data/Input	Output
1.	At most one symptom is entered.	"Please enter at least 2 symptoms for better results" message is displayed.
2.	2 symptoms are entered but disease found for individual symptom and combined result not found.	List of Diseases based on individual symptom displayed with message "provide more symptoms if possible."
3.	2 symptoms are entered and one or more diseases found	List of disease is displayed
4.	Symptoms are entered but not found any matching disease.	Message displayed "SORRY!! Not found any matching results. Ask question in FAQ to know about."

Conclusion

We have implemented most of the required functionalities for e-healthcare system. WeCare is the system which is useful to everyone to know about disease, symptoms, its treatments, hospitals, medical shops etc. As this is an healthcare system we can add more and more functionalities in future and update its features like diet plan module can be added, conversation between doctors can also be done in future.

Limitations

Users can not interact with doctors. Interaction between doctors is also not possible through this system. Location based search for hospitals and medical shops is limited to system database. If record is not present in system database then you are not able to found required details. Search disease based on symptoms is implemented using queries so results are not much accurate.

Future Extensions

We can add chat feature for interaction between user to doctor or between doctors(group chat). Video calling feature can also be implemented for better interaction between patient(user) and doctor.

For search functionality some ML classification algorithm like support vector machine, random forest or decision tree can be used for more accurate result of prediction.

Bibliography

To build this project we have taken references from the following websites,

- For framework used in development
  https://www.djangoproject.com/

- For CSS and other front end technologies
  https://www.w3schools.com/

- For Location based Python APIs
  https://pypi.org/project/geocoder/

- For Covid-19 Update Python API
  https://pypi.org/project/covid/
  https://api.covid19api.com

- For Google's News API
  https://pypi.org/project/GoogleNews/
  https://newsapi.org/s/google-news-api

- For Graphs.js
  https://www.chartjs.org/

- For Error solving
  https://stackoverflow.com/