

MEDEASE - *Healthcare Made Easy*

Minor Project

Nishant Srivastava - 14103033

Stuti - 14103056

Kanika Gupta - 14103271

Batch - B10

ABSTRACT

MEDEASE- Healthcare made Easy is a one stop solution to your health related queries and concerns. With practical and necessitous features like Symptom Checker, Doctor Recommendation, Hospital Locator and Medical History, it encompasses the needs of most people, the ones suffering with serious ailments to people who are not adept enough to physically visit a physician for a casual check-up or diagnosis.

Medease is a tool that can very much revolutionize how patients interact with their doctors and can bring about a massive transition in the healthcare industry.

OBJECTIVES

To develop an automated system which can do the following:

- Determine what disease a user is suffering from on the basis of what symptoms he/she is showing.
- Suggest doctors for consultation after the diagnosis of the disease.
- Display a route map to the doctor's practice.
- Record keeping of patient's medical/personal information.

SOFTWARE REQUIREMENTS

1. Python 2.7.12:

The programming language will be mainly used for the numerous frameworks that are available for making websites.

2. PyCharm 2016.2.3:

PyCharm is an Integrated Development Environment (IDE) used in computer programming, specifically for the Python language.

3. Django:

Django is a free and open-source web framework, written in Python, which follows the model–view–controller (MVC) architectural pattern.

4. Pyrenn:

A recurrent neural network toolbox for Python.

5. Numpy:

NumPy is the fundamental package for scientific computing with Python.

6. Pandas:

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. We used pandas to read our dataset from the excel sheet to provide input and output data to our neural network.

7. Openpyxl:

Openpyxl is a Python library for reading and writing Excel 2010 xlsx/xlsm/xltx/xltn files.

8. Database (SQLite):

To maintain the data required for the website in an organised tabular form.

9. Text editor (Textmate, Notepad++)

10. Google Chrome (Inspector)

11. HTML 5, CSS3

COMPONENTS OF MEDEASE

Medease is primarily divided into the following components :-

- **Symptom Checker**: An aid to people suffering from a disease in finding out what it is sitting at their home. The user is required to select a list of symptoms he is suffering from, from the given options, cough, headache, nausea for instance. After the user has selected all the symptoms, he/she is presented with a list of **possible diseases**.
- **Recommended Doctors**: After the user selects a disease, a list of certified doctors who can be consulted for the same on the basis of the diagnosed disease.
- **Hospital Locator**: The user will be displayed with a list of nearby hospitals and clinics along with a route map of the same that he can visit for treatment.
- **Medical History**: The user can create his own **profile** and enter all his/her **records** - the conditions he/she's suffering from, blood group, allergies, undergoing treatments and any other relevant information all at one place, saving the burden of carrying loads of paperwork every time he/she visits the doctor.

SYMPTOM CHECKER

THE IDEA:

Many a times we're suffering from a disease and we are hesitant in going to a doctor thinking of it to be a minor cold/cough and subject ourselves to 'self treatments', being ignorant to the fact that it can might as well be something very serious just to save the huge consulting fees and the pain of traveling.

MEDEASE provides it's users with an AI based symptom checker which suggests the user with a list of diseases he might be undergoing. Judging the possibilities, the user can than decide whether his condition requires visiting a doctor or is something too trivial, like a seasonal flu which can be treated at home.

BACKGROUND STUDY AND FINDINGS:

In order to build an accurate Symptom Checker we explored the field of Artificial Intelligence. Artificial intelligence is the branch of computer science concerned with making computers behave like humans and so to make our computer act like a doctor, AI seemed the best choice.

Artificial Learning is classified into two -

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

We, in our project, have implemented Supervised Learning.

Supervised learning is a data mining task of inferring a function from labeled training data. The training data consists of a set of training

examples. In supervised learning, each example is a pair consisting of an input object and the desired output value.

We have primarily used this type of Learning in our project.

Supervised Learning -

- Analytical learning
- Artificial neural network
- Backpropagation
- Bayesian networks
- Decision tree learning
- Learning Automata
- Nearest Neighbor Algorithm
- Support vector machines

We have implemented **Artificial Neural Network** in order to build our Symptom Checker .

Artificial Neural Networks are a computational approach which is based on a large collection of neural units loosely modeling the way a biological brain solves problems with large clusters of biological neurons connected by axons. Each neural unit is connected with many others, and links can be enforcing or inhibitory in their effect on the activation state of connected neural units. Each individual neural unit may have a summation function which combines the values of all its inputs together. There may be a threshold function or limiting function on each connection and on the unit itself such that it must surpass it before it can propagate to other neurons. These systems are self-learning and trained rather than explicitly programmed and excel in areas where the solution or feature detection is difficult to express in a traditional computer program.

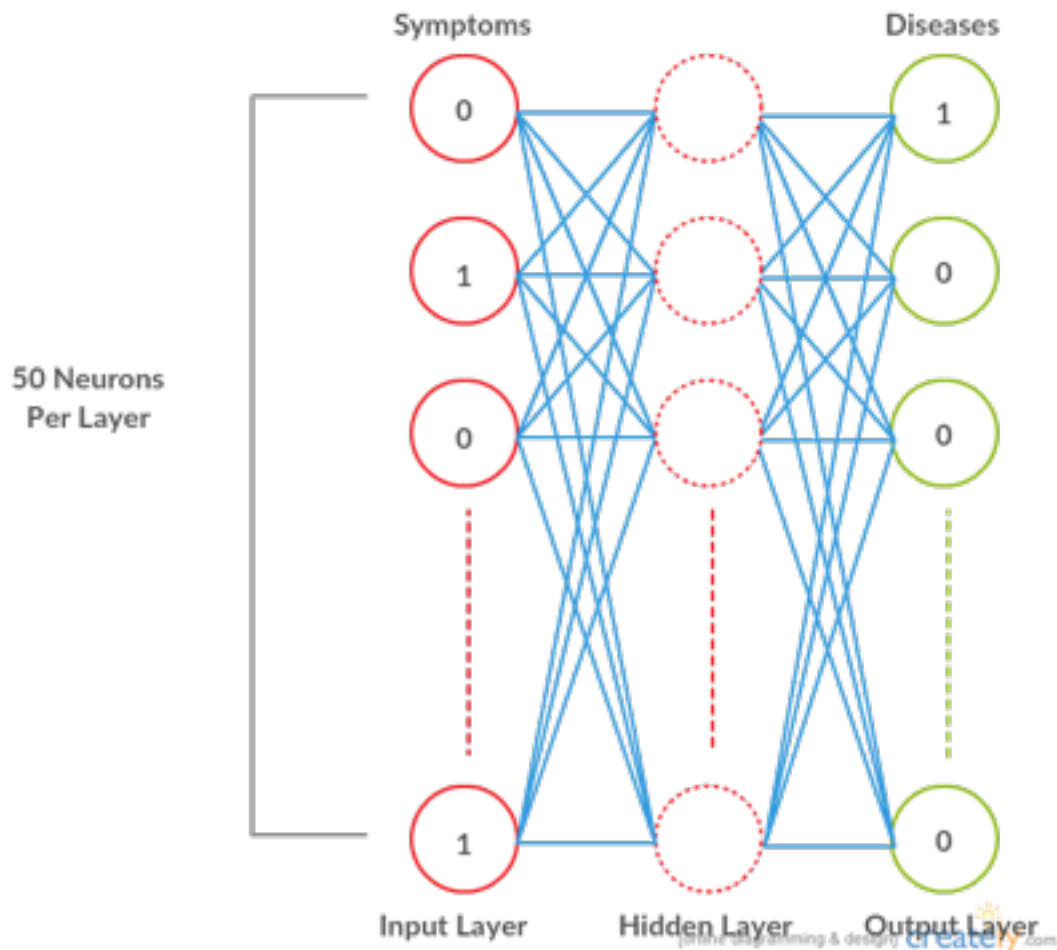
Artificial Neural Network -

- Feedforward neural network.
- Radial basis function (RBF) network.
- Kohonen self-organizing network.
- Learning vector quantization.
- Recurrent neural network.
- Modular neural networks.
- Physical neural network.

Our network is a **Feedforward neural network** - A feedforward neural network is an artificial neural network wherein connections between the units do not form a cycle. As such, it is different from recurrent neural networks. The feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

OUR IMPLEMENTATION

In our Neural Network we have 1 Input, Hidden and Output Layer each which we implemented using **pyrenn**. Each layer consists of 50 neurons. The 50 neurons in the input layer correspond to the symptoms which we are taking as Input from our user. The 50 neurons in the output layer correspond to 50 diseases in our dataset by which the symptom input from our user will be classified in. Following is a pictorial representation of what our network looks like:



For every symptom selected by our user, the value corresponding to that symptom in our dataset will be changed to 1 and the rest will remain 0. In the output layer the neurons with value 1 will correspond to the diseases that the symptom checker suggests the user might have.

Training Our Neural Network -

Our dataset consists of 50 diseases and a set of 50 symptoms.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	disease	aphtion	apynaw	exctio	ethene	backout	bradycardia	breath sound	chest tightness	chill	consciousness	cough	decreased bow	diarrhea	difficulty	distress	respir	drowsiness	dyapnea	facial persn	fatigue	feeling hopele	feeling			
2	acquired imm	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	adhesion	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	affect labile	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	Alzheimer's di	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0
6	anemia	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
7	aphasia	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	asthma	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
9	biliary colic	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	bigular disord	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
11	carcinoma pri	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
12	cholangiolitis	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
13	chronic elatrh	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
14	chronic kothri	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	chronic elatrh	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0
16	coronary arte	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	decubitus ulc	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	degenerative	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
19	deglutition di	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
20	dehydration	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
21	depressive di	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
22	diarrhoea	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	endocarditis	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
24	encephalopath	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	endocarditis	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
26	epilepsy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
27	failure heart	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
28	failure kidney	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
29	fibroad tumor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	gastritis	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	gout	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	hepatitis	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
33	hemia hater	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	hyperbilirubin	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
35	hypercholester	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
36	hyperglycemia	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	hyperthyroidis	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	hives	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	incontinence	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	infection urini	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
41	influenza	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	insufficiency r	0	1	1	1	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0
43	lymphoma	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	malignant mel	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	malignant mel	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
46	malignant tort	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	myocardial inf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

We have used **pandas** to read columns from the excel sheet to our python code and then training the data using pyrenn.

The network is trained as follows:

```
Python 2.7.12 (default, Oct 11 2016, 05:24:00)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.38)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 817.89023958927221, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 24.557082841315875, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.52968203153829541, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 0.0013478943527115862, '\tscale factor: ', 0.003)
('Iteration: ', 4, '\t\tError: ', 1.8732745612324225e-08, '\tscale factor: ', 0.0030000000000000003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 243.40321395881062, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 10.324286145933108, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.2938560734981356, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 0.00811858378969598345, '\tscale factor: ', 0.003)
('Iteration: ', 4, '\t\tError: ', 1.0512109557669351e-10, '\tscale factor: ', 0.0030000000000000003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 310.95832514531613, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 10.926994961753813, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.34228166934169135, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 0.0011473488179594858, '\tscale factor: ', 0.003)
('Iteration: ', 4, '\t\tError: ', 6.1268860694189888e-08, '\tscale factor: ', 0.0030000000000000003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 177.062099064635624, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 7.8677228843909335, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.25764969412100552, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 0.002424564857131778, '\tscale factor: ', 0.003)
>>>
```

10 diseases at a time

```
Python 2.7.12 Shell
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 64.91881424856093, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 5.3398611164403471, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.20399741533533736, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 0.0017374246139211908, '\tscale factor: ', 0.003)
('Iteration: ', 4, '\t\tError: ', 1.8764268296213787e-07, '\tscale factor: ', 0.0030000000000000003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 48.702821878730498, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 4.461885331691109, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.27159883154276765, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 0.0021961900619941258, '\tscale factor: ', 0.003)
('Iteration: ', 4, '\t\tError: ', 1.7968781335477383e-07, '\tscale factor: ', 0.0030000000000000003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 32.902223343364732, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 3.8287368397827017, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.28881859145631333, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 0.0024908917851273985, '\tscale factor: ', 0.003)
('Iteration: ', 4, '\t\tError: ', 2.5927798392187934e-07, '\tscale factor: ', 0.0030000000000000003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 36.411863619400572, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 3.3301194358626494, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.1328464375387535, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 0.0027371359620678801, '\tscale factor: ', 0.003)
('Iteration: ', 4, '\t\tError: ', 5.6689975053708335e-09, '\tscale factor: ', 0.0030000000000000003)
Termination Error reached
>>>
```

15 diseases at a time

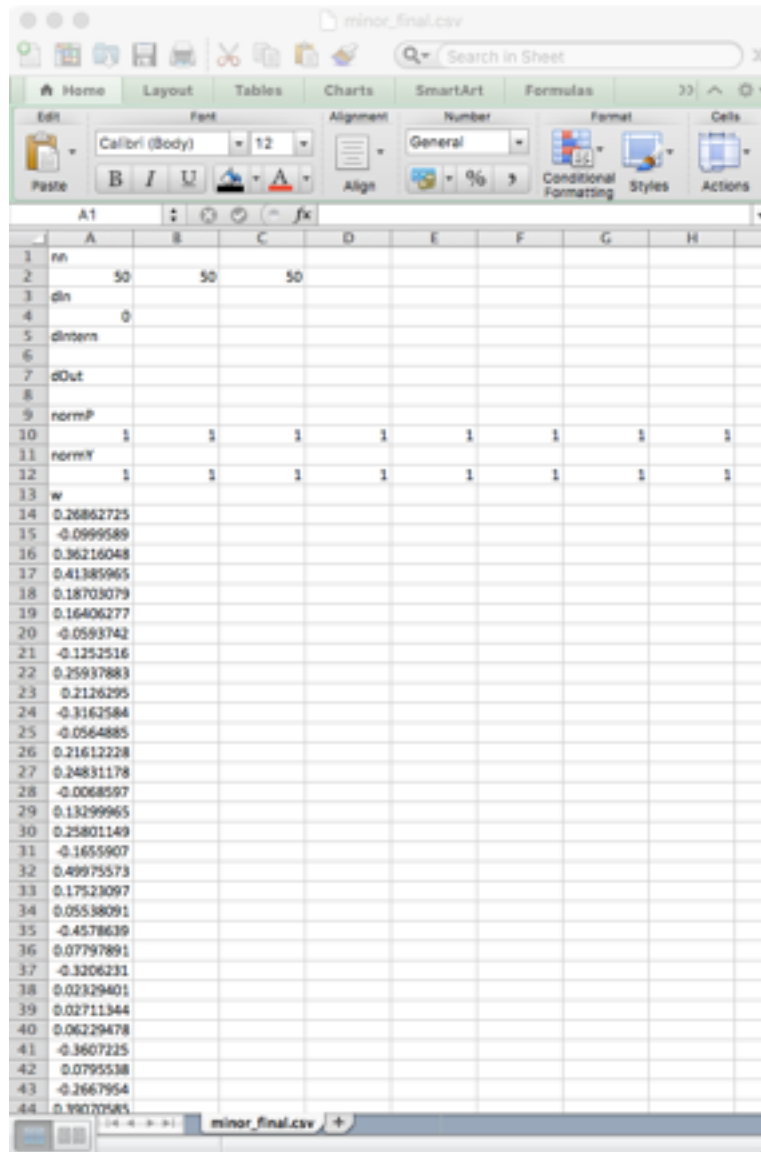
```
Python 2.7.12 Shell
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 16.251060714846439, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 2.3651282548315935, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.11486419526913504, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 0.00030952826747894166, '\tscale factor: ', 0.003)
('Iteration: ', 4, '\t\tError: ', 9.5358476147194649e-09, '\tscale factor: ', 0.0030000000000000003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 23.760218291291898, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 2.6978223347909399, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.22683995796424495, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 0.0011967848463361273, '\tscale factor: ', 0.003)
('Iteration: ', 4, '\t\tError: ', 7.510871782589788e-08, '\tscale factor: ', 0.0030000000000000003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 18.853945738367923, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 3.3351625656395725, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.28355180273989308, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 0.0024301794877581452, '\tscale factor: ', 0.003)
('Iteration: ', 4, '\t\tError: ', 5.1220485165622792e-08, '\tscale factor: ', 0.0030000000000000003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 28.123179240604259, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 4.5162871142727159, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.57858917290013101, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 0.011906646574213675, '\tscale factor: ', 0.003)
('Iteration: ', 4, '\t\tError: ', 3.421832334034463e-06, '\tscale factor: ', 0.0030000000000000003)
Termination Error reached
>>>
```

20 diseases at a time

```
Python 2.7.12 Shell
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 6.827078659887809, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 0.65846646038256614, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.025310262794565883, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 3.8322002551475819e-05, '\tscale factor: ', 0.003)
('Iteration: ', 4, '\t\tError: ', 4.1740667331743717e-10, '\tscale factor: ', 0.0030000000000000003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 5.6742242537594911, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 0.49271759337918002, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.0050012916138092581, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 7.2764388738272931e-07, '\tscale factor: ', 0.003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 4.7424483831006067, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 0.40217180896171432, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.0058001122621104849, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 7.5783975681893208e-07, '\tscale factor: ', 0.003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 4.5538094833568234, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 0.17801809667888663, '\tscale factor: ', 0.3)
('Iteration: ', 2, '\t\tError: ', 0.0053139885558174, '\tscale factor: ', 0.03)
('Iteration: ', 3, '\t\tError: ', 4.93092972427254e-07, '\tscale factor: ', 0.003)
Termination Error reached
>>>
== RESTART: /usr/local/lib/python2.7/site-packages/examples/Minor_final.py ==
('Iteration: ', 0, '\t\tError: ', 2.0623081243737218, '\tscale factor: ', 3.0)
('Iteration: ', 1, '\t\tError: ', 0.13298900885780537, '\tscale factor: ', 0.3)
>>>
```

5 diseases at a time

To get the most accurate results and reduce the error. This is our trained network.



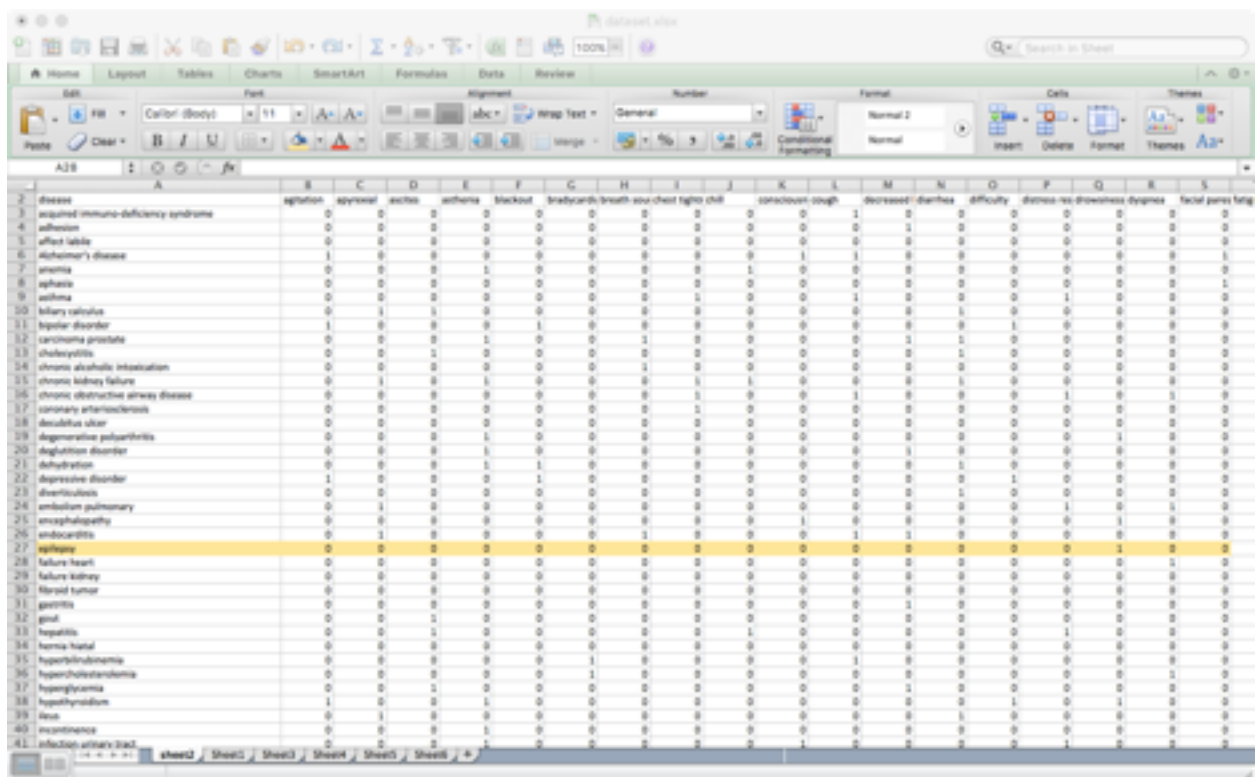
	A	B	C	D	E	F	G	H
1	nn							
2		50	50	50				
3	gln							
4		0						
5	dlntern							
6								
7	dOut							
8								
9	normP							
10		1	1	1	1	1	1	1
11	normY							
12		1	1	1	1	1	1	1
13	w							
14		0.268862725						
15		-0.0999589						
16		0.36216048						
17		0.41385965						
18		0.18703079						
19		0.16406277						
20		-0.0593742						
21		-0.1252516						
22		0.25937883						
23		0.2126295						
24		-0.3162584						
25		-0.0564885						
26		0.21612228						
27		0.24831178						
28		-0.0068597						
29		0.13299965						
30		0.25801149						
31		-0.1655907						
32		0.49975573						
33		0.17523097						
34		0.05538091						
35		-0.4578639						
36		0.07797891						
37		-0.3206231						
38		0.02329401						
39		0.02711344						
40		0.06229478						
41		-0.3607225						
42		0.0795538						
43		-0.2667954						
44		0.39070585						

Using Our Neural Network To Get Output -

While taking input from a user, we used Django to translate those inputs into Binary and then used Openpyxl to write to our Excel sheet 'test.xlsx' which served as our test dataset.

This test data then gives us the required outputs on the basis of the training that we did earlier.

This output is then displayed to our user by converting it back from Binary using Django.



The screenshot shows an Excel spreadsheet titled 'dataset.xlsx'. The spreadsheet has columns A through S. Column A lists various diseases, and columns B through S contain binary values (0 or 1) representing different symptoms or features for each disease. The diseases listed are:

- 2. disease
- 3. acquired-immuno-deficiency syndrome
- 4. adhesion
- 5. affect-labile
- 6. alzheimer's disease
- 7. anemia
- 8. aphasia
- 9. asthma
- 10. biliary calculus
- 11. bipolar disorder
- 12. carcinoma prostate
- 13. cholecystitis
- 14. chronic alcoholism intoxication
- 15. chronic kidney failure
- 16. chronic obstructive airway disease
- 17. coronary arteriosclerosis
- 18. decubitus ulcer
- 19. degenerative polyarthritis
- 20. deglutition disorder
- 21. dehydration
- 22. depressive disorder
- 23. diverticulitis
- 24. embolism pulmonary
- 25. encephalopathy
- 26. endocarditis
- 27. epilepsy
- 28. failure heart
- 29. failure kidney
- 30. fibroid tumor
- 31. gastritis
- 32. gout
- 33. hepatitis
- 34. hernia hiatal
- 35. hyperbromidemia
- 36. hypercholesterolemia
- 37. hyperglycemia
- 38. hypothyroidism
- 39. ileus
- 40. incontinence
- 41. infectious mononucleosis

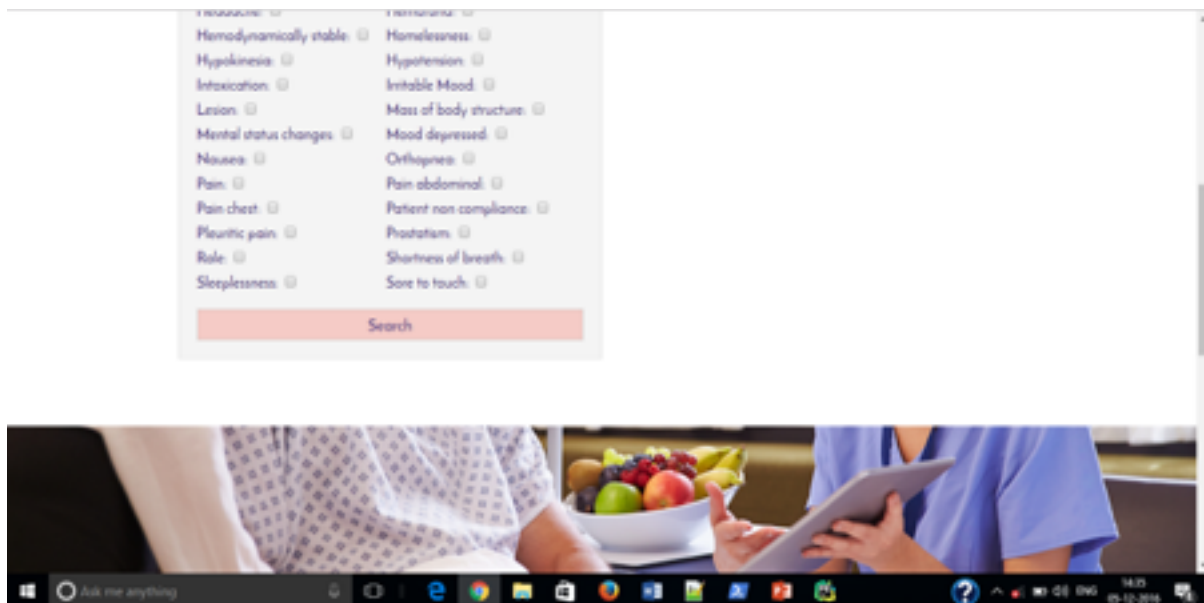
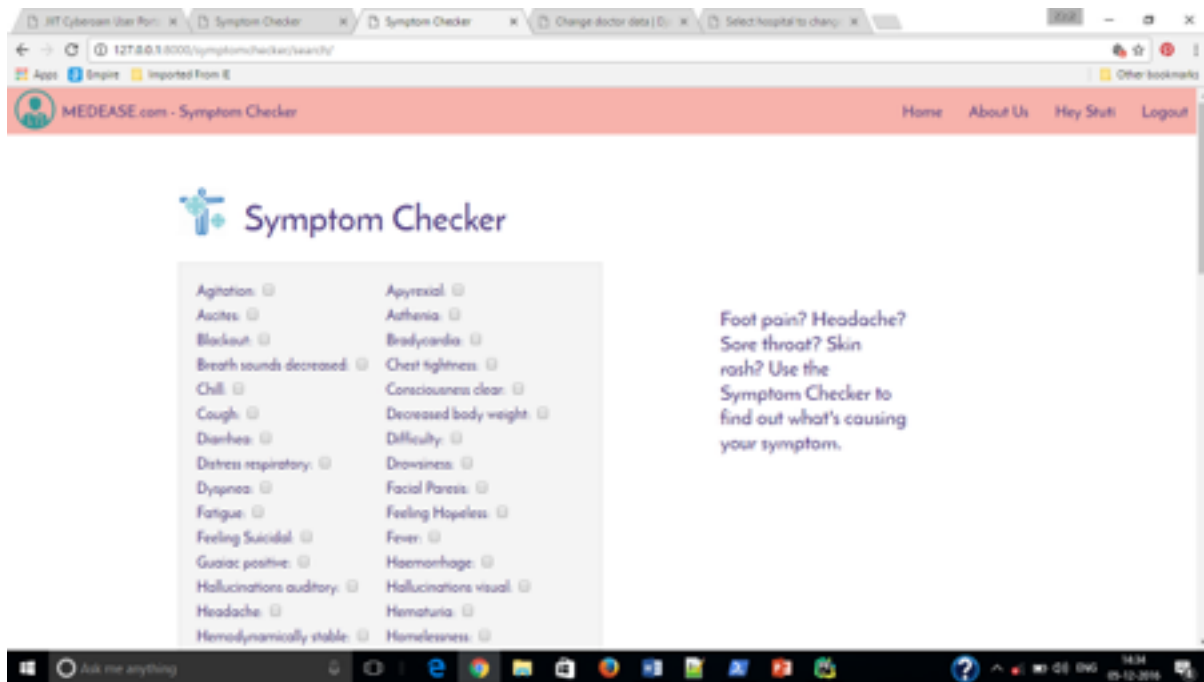
The binary values are 0 or 1, indicating the presence or absence of a symptom for each disease. For example, for 'epilepsy' (row 27), the values are 0 for agitation, apyrexia, icterus, anorexia, blackout, bradycardia, breath-slow, chest-tight, chill, consciousness-rough, decreased, diarrhea, difficulty, dizziness, no-drowsiness, dyspnea, facial-paresis, and fatigue.

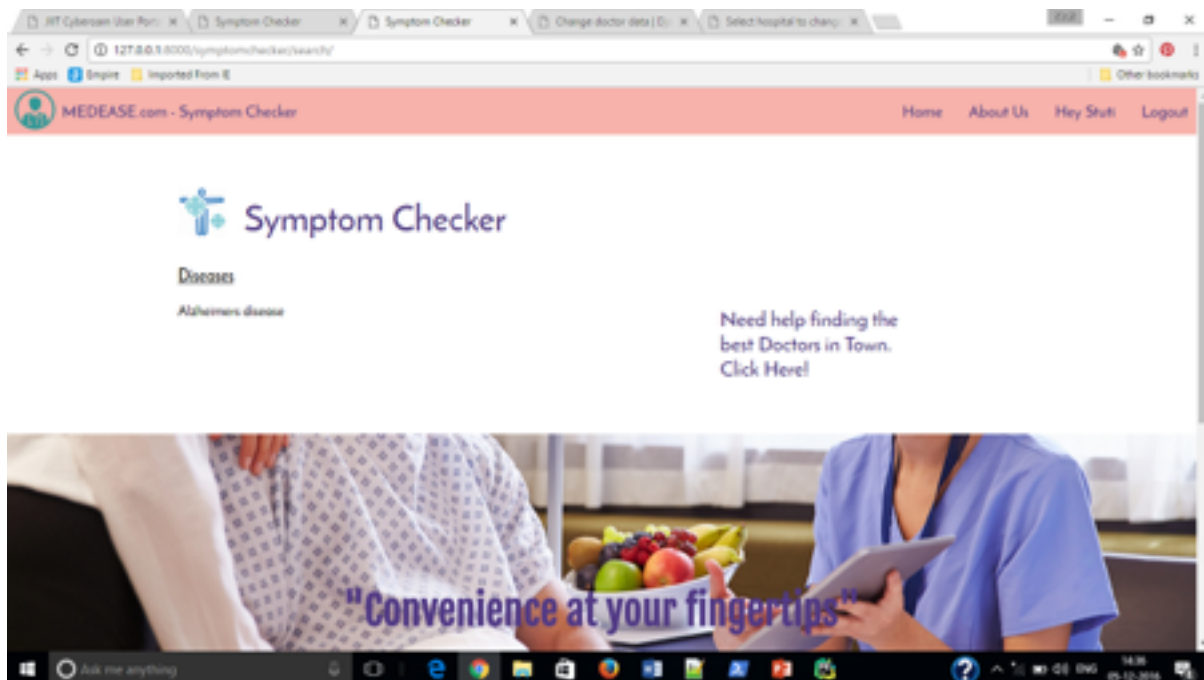
Below are the outputs at the output neurons which are then normalized to give values 0/1.

```
Python 2.5.2 Shell
0.7848237708531
-0.598599411952
-0.8135281326654
0.85249246053288
0.15823473790
-0.8996939811635
-0.891201437659
-0.8377241905656
0.806577767739
-0.8701812219098
0.8465385373294
-0.80361771248281
0.80298144698372
0.8158802416795
-0.808675482938978
-0.8515189264798
0.8937115540996
-0.117584989648
-0.185267839512
0.844659187675
0.8232372246984
-0.8897812209642
0.8854781419653
0.718791205722
0.8475396831161
-0.8296373355649
-0.8775399713952
-0.8885681547575
-0.881442278863
0.8438724354799
-0.8887486988484
-0.842558549643
0.8829936367142
-0.88298324389585
-0.8359355205683
0.88441858838125
-0.8491189556874
0.8813965336721
0.8287668261783
0.123953819932
```

[illegible]

After integration with Django, the UI of the Symptom Checker is as follows:





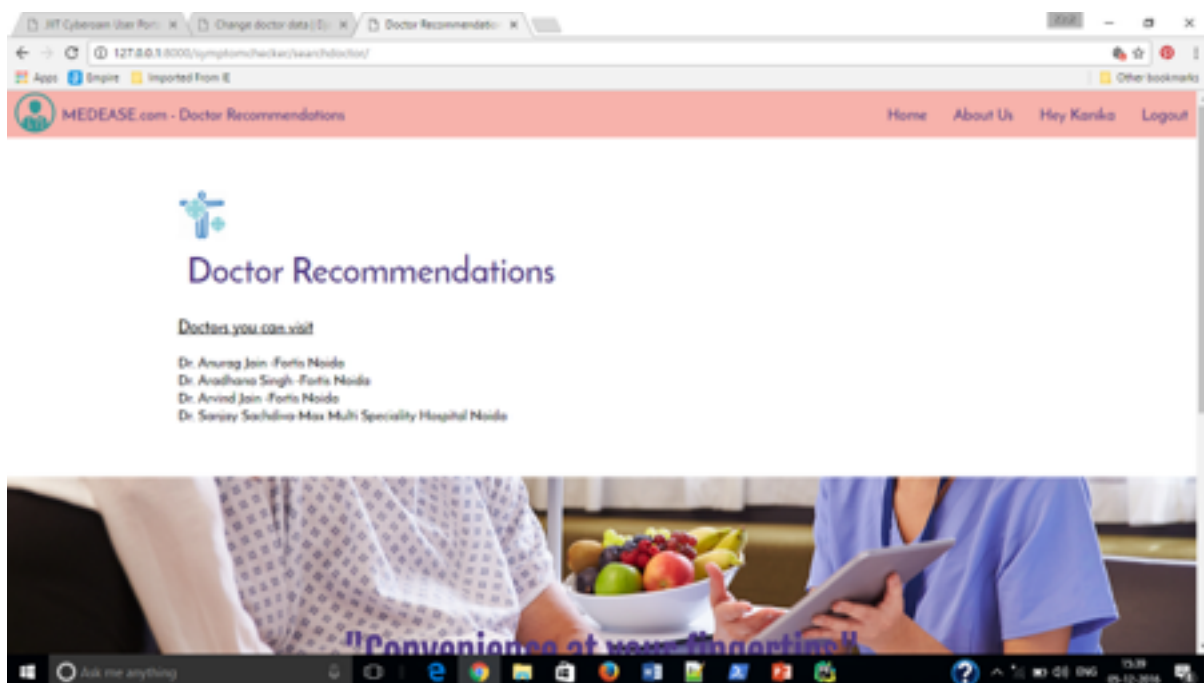
SUGGESTED DOCTORS

The table from which we retrieve our data to suggest the doctors consists of the following fields -

Day	Hours
Monday	2:00 pm to 5:00 pm
Tuesday	2:00 pm to 5:00 pm
Wednesday	2:00 pm to 5:00 pm
Thursday	2:00 pm to 5:00 pm
Friday	2:00 pm to 5:00 pm
Saturday	2:00 pm to 5:00 pm
Sunday	none

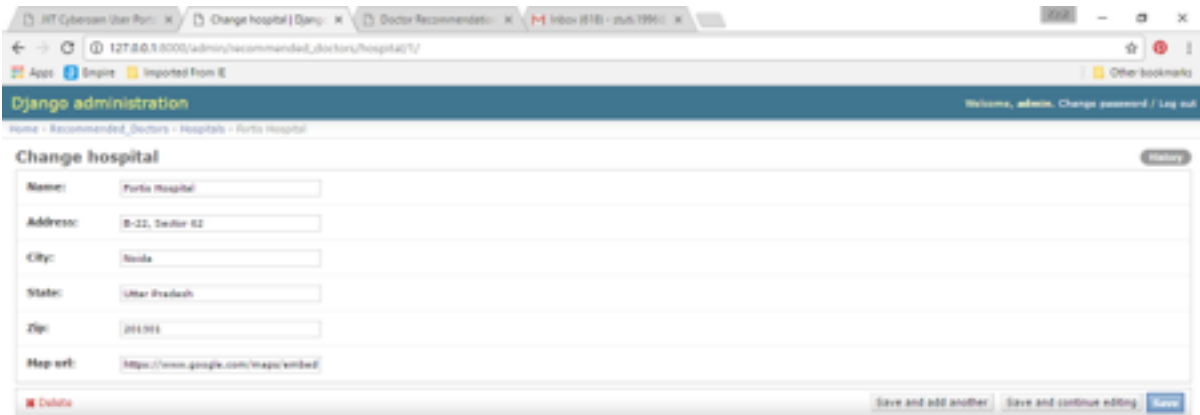
The doctors are suggested on the basis of the type of disease that resulted from our Symptom Checker thus seamlessly bounding the two applications.

The list of diseases obtained from our Symptom Checker is first mapped to its type which then helps us to map it further to the doctor that the person should consult using the field “Type” in our Doctor_data model.



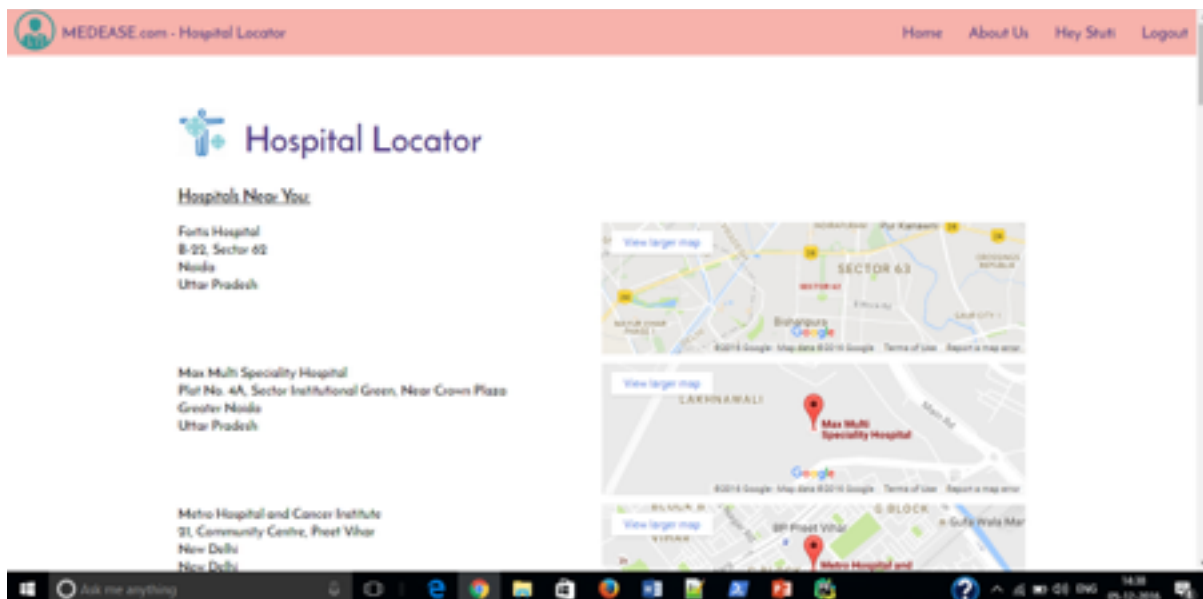
HOSPITAL LOCATOR

The table from which we retrieve our data to locate the Hospitals consists of the following fields -



The screenshot shows a web browser window with multiple tabs. The active tab is titled 'Change hospital | Django'. The address bar shows the URL '127.0.0.1:8000/admin/recommended_doctors/hospital/1/'. The page header indicates 'Django administration' and 'Welcome, admin. Change password / Log out'. The main content area is titled 'Change hospital' and contains a form with the following fields: Name (Fortis Hospital), Address (B-22, Sector 62), City (Noida), State (Uttar Pradesh), Zip (201301), and Map url (https://www.google.com/maps/embed/). At the bottom of the form are buttons for 'Delete', 'Save and add another', 'Save and continue editing', and 'Save'.

To help a user find the nearest hospitals the user must first be logged in and fill up his personal information using our medical profile app. The address , city and state fields entered by the user is used to locate the nearby hospitals .



Medical History

The table from which we retrieve our data to display the medical history of a user consists of the following fields -

The screenshot shows a web application interface for changing user details. The browser window has several tabs open, including 'RT Cybercam (User Port)', 'Change user_details (1)', 'Doctor Recommendation', and 'Inbox (118) - stub.1994'. The address bar shows the URL '127.0.0.1:8000/admin/profile_user/user_details/1/'. The page title is 'Change user_details'. The form contains the following fields:

- First name:
- Last name:
- Email id:
- Date of Birth: Today
- Notice: You are 5.5 hours ahead of server time.
- Password:
- Gender:
- Address:
- City:
- State:
- Zip:
- Phone:
- Blood group:
- Drug allergy:
- Addict habit:
- Psych ill:
- Hypertension:
- Diabetes:
- Asthma:
- Bronchitis:
- Fits:
- Any congenital:
- Surgery accident:
- Drug intake:
- Other:

At the bottom of the form, there are three buttons: 'Delete', 'Save and add another', and 'Save and continue editing'. The 'Save' button is also visible.

This application is for the user to keep a track of his/her Medical history as it is very crucial for diagnosis of any other ailment which the user may contact. The choices entered in the various check boxes and character fields are stored in their respective databases and can be edited as per the user's convenience.

The screenshot shows a web browser window with the URL 127.0.0.1:8000/user/profile/. The page title is "Medical Profile". On the left, there is a sidebar with the following menu items: "Medical Information", "Edit Profile", "Edit Medical Information", and "Change Password". The main content area displays the following personal information:

First Name:	Stuti
Last Name:	Kumar
D.O.B:	July 18, 1996
Gender:	Female
Email Address:	stuti.1996@gmail.com
Address:	Sector-33
City:	Noida
State:	Uttar Pradesh
Zip:	201301
Contact Number:	9540250576

The screenshot shows the same web browser window, but the "Medical Information" menu item is selected. The main content area displays the following medical information:

Blood Group	
Any History of Allergic to Drug:	False
History of Addiction	
Psychiatric Illness:	False
Hypertension	False
Diabetes Mellitus:	
Bronchial Asthma:	
Allergic Bronchitis:	False
Convulsions or Fits:	False
Any Congenital Diseases:	False
History of any major Surgery or Accident:	0
History of any Drug Intake :	False
Other History (If Any):	

RT Cybercam User Profile Site administration User Profile

127.0.0.1:8000/user/profile/

Medical Profile

Personal Information	First Name: Stuti Last Name: Kumar Email-Id: stuti.1996@gmail.com Gender: <input type="text"/>
Medical Information	Address: <input type="text"/>
Edit Profile	City: <input type="text"/>
Edit Medical Information	State: <input type="text"/>
Change Password	Zip: <input type="text"/>
	Phone Number: <input type="text"/>
	<input type="button" value="Submit"/>

RT Cybercam User Profile Site administration User Profile

127.0.0.1:8000/user/profile/

Medical Profile

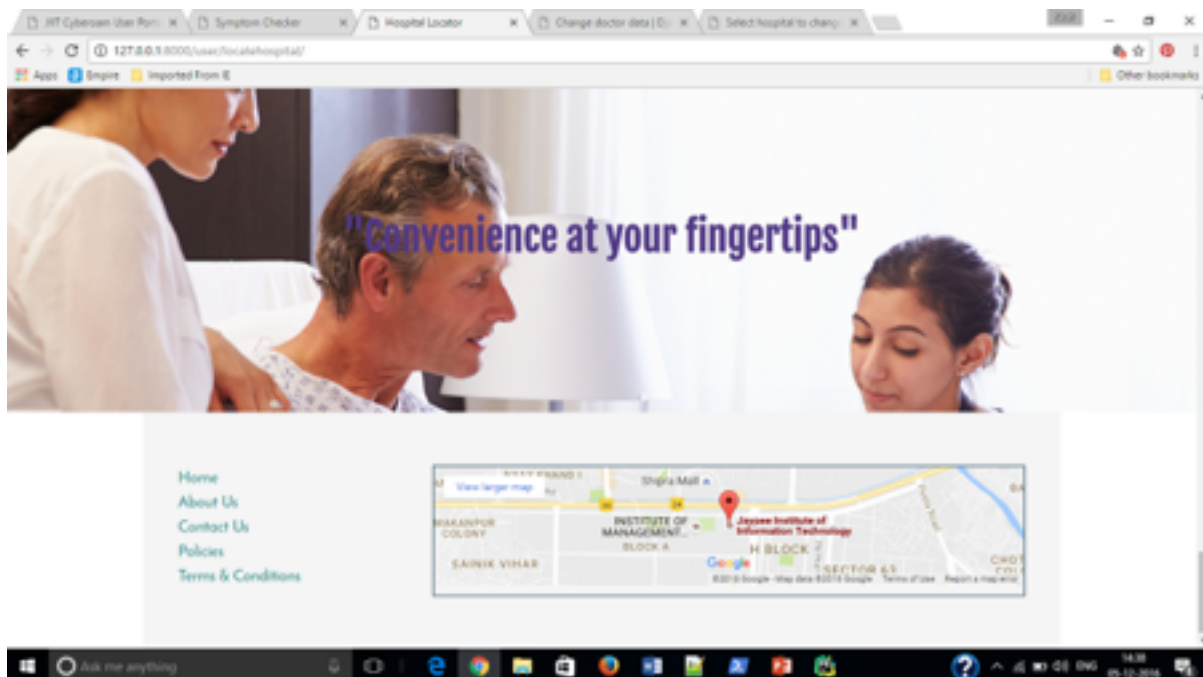
Personal Information	Blood Group: <input type="text"/>
Medical Information	Any History: <input type="checkbox"/> Addiction history: <input type="checkbox"/> Any illness: <input type="checkbox"/> Hypertension: <input type="checkbox"/> Diabetes: <input type="checkbox"/> Asthma: <input type="checkbox"/> Bronchitis: <input type="checkbox"/> Fits: <input type="checkbox"/> Congenital Diseases: <input type="checkbox"/> Any previous surgeries/accidents: <input type="text"/>
Edit Profile	Any Drug Intake: <input type="checkbox"/> Addiction history: <input type="text"/>
Edit Medical Information	<input type="button" value="Submit"/>
Change Password	

RT Cybercam User Profile Site administration User Profile

127.0.0.1:8000/user/profile/

Medical Profile

Personal Information	
Medical Information	
Edit Profile	Old Password: <input type="text"/>
Edit Medical Information	New Password: <input type="text"/>
Change Password	<input type="button" value="Submit"/>



REFERENCES

Online Documentations and Tutorials:

1. <https://www.djangoproject.com/>
2. www.tutorialspoint.com/django
3. <https://www.youtube.com/watch?v=qgGlqRFvFFk>
4. <https://www.codecademy.com/learn/web>
5. www.w3schools.com/html/
6. <https://www.youtube.com/watch?v=bx2T-V8XR&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU> - Neural Networks Demystified
7. pyrenn.readthedocs.io/en/latest/create.html
8. <https://github.com/yabata/pyrenn>
9. <https://openpyxl.readthedocs.io/en/default/>
10. <https://docs.scipy.org/doc/numpy/reference/>
11. <http://pandas.pydata.org/pandas-docs/stable/>

Research Papers:

- *Artificial Neural Networks (2012)*, G. Kumar JHA, Indian Agricultural Research Institute - <https://scholar.google.com/scholar?oi=bibs&cluster=14779615864438299204&btnI=1&hl=th>
- *An Evolutionary Approach: Analysis of Artificial Neural Networks* - http://www.ijetae.com/files/Volume2Issue1/IJETAe_0112_30.pdf
- *Utilization of Neural Network for Disease Forecasting* - <http://www.statistics.gov.hk/wsc/IPS063-P4-S.pdf>
- *Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture* - <http://ijettjournal.org/volume-3/issue-6/IJETT-V3I6P206.pdf>

