

A Report On

Personalized Health Assistant

Submitted by

A.Namratha Deepthi(14IT106)

Parvathi.M.H(14IT130)

Sneha Jhaveri(14IT143)

Srivalya Elluru(14IT243)

in partial fulfillment for the award of the degree

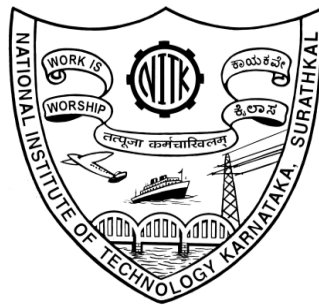
of

Bachelor of Technology

In

Information Technology

At



Department of Information Technology

National Institute of Technology Karnataka, Surathkal.

September 2016

Department of Information Technology

ABSTRACT

Statistics show that more than 70% of the internet users look up for health information online. The commonly researched topics include specified diseases or conditions, treatments or procedures and doctors or health professionals. Most of the online health seekers take the help of search engines for their doubts and complain on the lack of a proper website. Though there is abundant information on the internet, critics often question the quality of online health information. Our project thus, mainly aims at being a one stop solution to the health problems of the users. This is done by predicting the possible diseases he/she may be suffering from based on a given set of symptoms.

This project also aims at bridging the information gap between patients and doctors in Bangalore by adopting Text Analytics. It helps in providing a platform that enables the public to locate a doctor with just their symptoms as inputs, rather than scientific or professional medical terms. As a result, a list of doctors' information such as the name of their clinic and map location are displayed. In today's world, Computer Aided Decision Support System plays a major task in medical field. Data mining provides the methodology and technology to alter the available mounds of data into useful information for decision making. By using data mining and natural language processing techniques, the time taken for the prediction of a disease is relatively less and the results are more accurate. Incorporation of the above techniques in our website has helped us in satisfying the general user requirements to a good extent.

In this way our website offers a widespread access to health information, along with the advantages of information tailoring and anonymity. We aim at ensuring that our website is helpful to the users for learning about the symptoms of their diseases, managing the symptoms and understanding the treatments.

TABLE OF CONTENTS

| | |
|--|----|
| 1. Introduction..... | 1 |
| 1.1 Scope of the work..... | 2 |
| 1.2 Usage Scenarios..... | 2 |
| 2. Requirements Analysis..... | 3 |
| 2.1 Functional requirements..... | 3 |
| 2.2 Non Functional Requirements..... | 3 |
| 2.2.1 Performance/Time..... | 3 |
| 2.2.2 Usability..... | 3 |
| 2.2.3 Security..... | 4 |
| 2.2.4 Reliability..... | 4 |
| 3. Conceptual Models..... | 5 |
| 3.1 Entity Relationship Diagram..... | 5 |
| 3.2 Relational Schema..... | 6 |
| 3.3 Normalization..... | 6 |
| 3.3.1 First Normal Form..... | 6 |
| 3.3.2 Second Normal Form | 6 |
| 3.3.3 Third Normal Form..... | 7 |
| 4. System Design..... | 7 |
| 4.1 Design Goals..... | 7 |
| 4.2 System Architecture..... | 8 |
| 4.3 Detailed Design Methodology..... | 9 |
| 4.3.1 Data Collection..... | 9 |
| 4.3.2 Pre-processing | 9 |
| 4.3.3 Normalized tf-idf computation..... | 10 |
| 4.3.4 Heart Disease Prediction..... | 12 |
| 4.3.5 Geo-coding of postal codes..... | 13 |
| 4.3.6 Development environment..... | 13 |
| 5. Results and Analysis..... | 14 |
| 6. Conclusion and Future Scope..... | 19 |
| 7. References..... | 20 |

1. INTRODUCTION

In today's world, the most commonly researched topics on the internet include specified diseases or conditions, treatments or procedures and doctors or health professionals. Most of the online health seekers take the help of search engines for their doubts and complain on the lack of a proper website. Though there is abundant information on the internet, critics often question the quality of online health information. Hence, there is a dire need for a one stop solution to the health problems of the users. With numerous advancements in the realm of healthcare, the problem of “too many” still persists in the current age. Patients struggle to find the right medical professional for their symptoms near their location. With the above problems as inspiration, we have developed a solution that gives patients the possible diseases that you might have based on their symptoms, along with the list of doctors specialized for those diseases in their locality. The proposed solution is limited to the city of Bangalore in terms of suggestion of doctors.

In India, one fifth of the deaths are from coronary heart diseases. According to statistics, by the year 2020, it will account for one third of all the deaths. The growth of heart diseases is dependent on a number of interlinked factors such as ageing, changing lifestyles, bad eating habits and rapidly evolving socio-economic determinants like access to healthcare. Thus, as an extension to our project, we have incorporated a heart disease predictor to help patients with adequate health information in checking the possibility of them suffering from a heart disease and getting a general idea about their heart health.

To summarize, our website helps in minimizing users' time, effort and anxiety when finding the possible diseases for their symptoms and the right medical help.

1.1 Scope of the work

The main activities in scope of this work are:

- Understanding and analyzing the requirements of the healthcare web application.
- System Design
- Implementation
- Unit Testing
- Demo

1.2 Usage Scenarios:

Our website can be used in case of medical emergencies when the users can have a quick search for the specified symptoms and look for the immediate precaution/medicine to be taken. The website also provides a complete guide for a healthy living. Certified doctors are allowed to register to our website, whose names will be reflected in the patients' search. The three main functionalities are 'Symptom checker', 'Find a doctor', 'HeartDiseasePredictor'.

The main usage scenarios are:

- Whenever a person falls sick and is in need of quick medical aid, he would resort to this website and enter his query with symptoms and doctor is recommended.
- Users can search for doctors based on specialization and/or location.
- Patients can provide required health parameters to predict if they have a heart disease or not.

2. REQUIREMENT ANALYSIS

2.1 Functional Requirements

- This web application shall allow users to register themselves and have their personal accounts with a short bio about themselves.
- The users' (patients) shall be allowed to enter their symptom queries and find the ailment for the same.
- The users' can also find doctors based on specialization and/or location.
- A heart disease predictor will be available which predicts if a patient has heart disease based on certain parameters. Some of the parameters will be obtained after a detailed checkup by the doctor.
- Users' personal medical history shall also be maintained.

2.2 Non functional Requirements

2.2.1 Performance/Time

The query processing and the retrieval of the result should take around 5 seconds in total. Since the queries entered by users' can vary, the processing time can vary slightly.

2.2.2 Usability

Usability is a crucial point in the system. The web interface should be user friendly such that anyone using it for the first time is able to use all the functionalities and it should solve his purpose.

2.2.3 Security

Secure login feature should be implemented so as to make sure that no user can access other users' medical history or personal details.

2.2.4 Reliability

This system must ensure that the details of a disease given for a particular set of symptoms and the medicines suggested for that disease are accurate.

3. CONCEPTUAL MODELS

3.1 Entity-Relationship Diagram

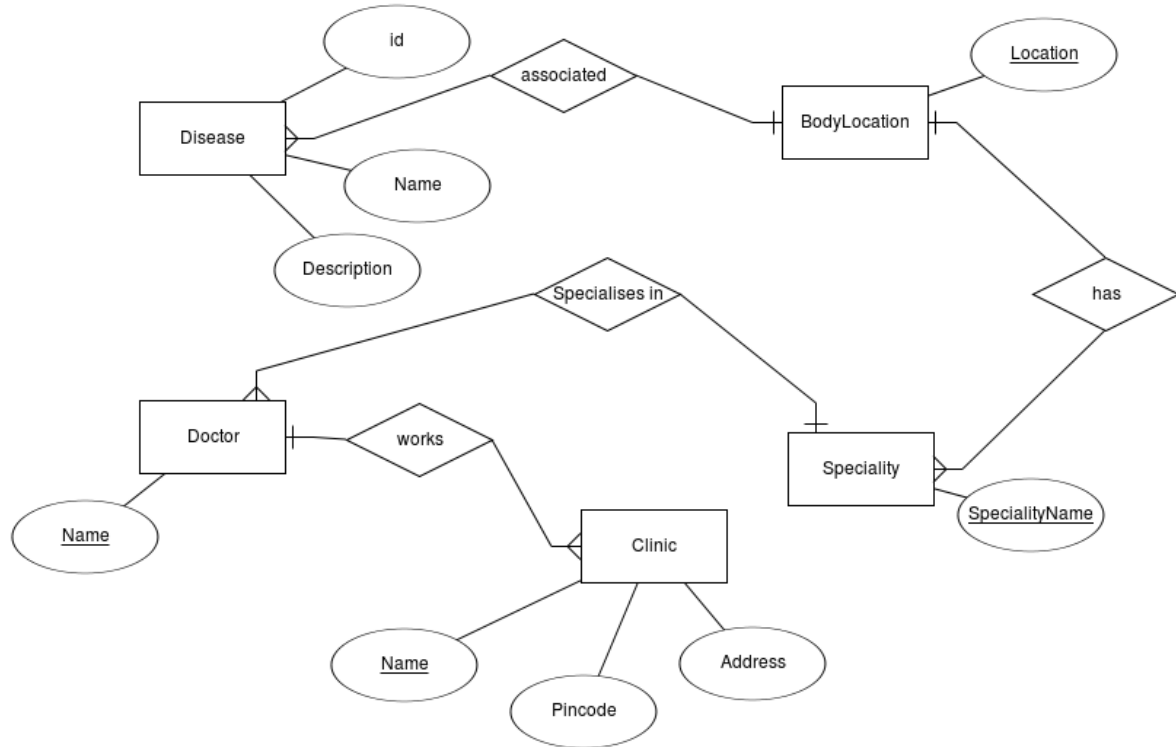


Figure 1: Entity Relationship Diagram

The above diagram shows the various entity sets, entity attributes and the relationships between the entities along with the cardinalities and participation of the relationships.

3.2 Relational Schema

The ER diagram represents the conceptual level of database design. Meanwhile, the relational schema is a logical level for the database design.

The transformation from the Entity-Relationship Diagram to the relation schema is quite straightforward.

DiseaseSymptoms :

| | | | |
|-----------|--------------|---------|----------|
| DiseaseID | BodyLocation | Disease | Symptoms |
|-----------|--------------|---------|----------|

body_speciality :

| | |
|---------------|------------|
| body_location | Speciality |
|---------------|------------|

Doctors :

| | | | | |
|------------|------------|------------|---------|---------|
| DoctorName | Speciality | ClinicName | Address | PinCode |
|------------|------------|------------|---------|---------|

3.3 Normalization

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. In our database, normalization was carried out keeping functional dependencies and candidate keys in mind. A step-wise normalization process has been followed, till all the redundant data and anomalies have been eliminated. The description of the different normalization forms is given below :

3.3.1 First normal form (1NF) - A relation is said to be in its first normal form if it does not contain any multi-valued or composite attributes and thus, each row can be uniquely identified with the help of a primary key. The outcome of first normal form is that a primary key can be identified in each table using a minimal super key and also the issues regarding composite and multi-valued attributes will be resolved.

The limitation of the first normal form is that, data redundancy increases, as there will be many columns with the same data in multiple rows but each row as a whole will be unique.

3.3.2 Second Normal Form (2NF) - As per the Second Normal Form there must not be any partial dependency of any column on the primary key. It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. To convert a table into its second normal form we need to ensure that, repeating column values should be removed and maintained in a separate table. This way updation can be done only once in the new table

rather than all entries in the first table. The rule to be kept in mind is that a foreign key must be on the N side which would otherwise result in the occurrence of multiple-values in a column. Next we need to identify a prime attribute (part of candidate key that determines anything else), called the partial dependency, and eliminate it. Finally, a foreign key should be made use of on the many side.

The limitation of the Second Normal Form is that it suffers Update Anomalies, and to in order to handle those scenarios, a Third Normal Form is made use of.

3.3.3 Third Normal Form (3NF)

For a table to be in its third normal form, every non-prime attribute of the table must be dependent on primary key, or we can say that, there should not be the case that a non-prime attribute is determined by another non-prime attribute. So these rows should be removed from the table and also the table must be in its Second Normal form. After reaching 3NF, database is considered to be normalized.

In our database, normalization has been applied on to the Doctors table as it has the redundancy and anomaly issues. The table DiseaseSymptoms need not be normalized as it is used only to query and retrieve data and is never used for updation.

The tables in the database after updation are as follows :

clinic_add :

| | |
|------------|---------|
| ClinicName | Address |
|------------|---------|

clinic_lat_long :

| | | |
|------------|----------|-----------|
| ClinicName | Latitude | Longitude |
|------------|----------|-----------|

clinic_pin :

| | |
|------------|---------|
| ClinicName | PinCode |
|------------|---------|

doc_clinic :

| | |
|------------|------------|
| DoctorName | ClinicName |
|------------|------------|

doc_speciality :

| | |
|------------|------------|
| DoctorName | Speciality |
|------------|------------|

4. System Design

The overall workflow of this project can be divided into three sections as shown below.



Figure 2: Workflow of Project

4.1 Design Goals

- This website should be responsive across all devices.
- The design should follow model view controller architecture with loose coupling by using Django with Python, MySQL database and with html, css3 and materialize css as front end.
- The code should be modular and extensible and meet all the functional and non-functional requirements mentioned in the above section.

4.2 System Architecture

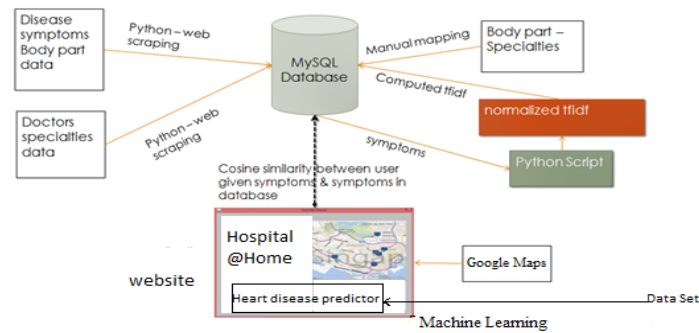


Figure 3: System Architecture

4.3 Detailed Design Methodology

4.3.1 Data Collection

Data Source:

The website requires three sets of data namely a set of symptoms for a particular disease, body location mapped to the disease, the doctors available in the city of Bangalore for the speciality. All the collected data is stored in a *MySQL* database for the purpose of easy retrieval and querying. To collect the disease-symptoms information, Beautiful Soup library available in python has been used. Web Scrapping was done from the following URL: <https://medlineplus.gov/healthtopics.html>.

The output from the scraper is loaded into the DiseaseSymptoms table.

A similar method was employed for obtaining the data about Doctors residing in Bangalore. The URL used for the same is : <http://www.medecure.com/doctor-in-bangalore.html> . The collected data was loaded into the Doctors table which was normalised as shown earlier.

A Body Location and Specialty mapping was done manually. The corresponding entries are available in body_speciality table. The table shows that a body location can be mapped to

multiple specialties. This was done to list doctors from different specialties for diseases on some specific body locations.

For the heart disease prediction, the dataset was obtained from the UCI Machine Learning repository-"processed.cleveland.data".

4.3.2 Pre-processing

To rank the similarity of user given symptoms to symptoms of any disease in the database, a natural language processing technique - '*cosine similarity*' has been used. In order to use cosine similarity, the documents (in our case the symptoms) have to be represented as vectors. The representation of a set of documents as vectors in a common vector space is known as vector space model and is a fundamental technique used for scoring documents on a query. In this 'bag of words' vector representation of documents, each document is represented as shown below.

$D_i = (W_{1i}, W_{2i}, \dots, W_{ni})$ where W_{1i}, W_{2i} are weights of terms(words) within the document. In our case Weights are computed using 'Term Frequency - Inverse Document Frequency' (tf-idf) given by the formula.

$$tfidf = (1 + \log(tf)) \times \log\left(\frac{N}{n}\right)$$

Where,

$(1 + \log(tf))$ = a variant of term frequency¹, $\log\left(\frac{N}{n}\right)$ = the inverse document frequency,

n = the document frequency of a term, N = the total number of documents

The first factor, term frequency (tf) accounts for the frequency of a term with in a document and inverse document frequency (idf), the frequency of a term across multiple documents. Terms that are occurring in all documents will have low tfidf scores as n approaches N and $\log(N/n)$ approaches 0. This will help to demote common words (and not stop words) that occur in most documents. Similarly, if a term is highly frequent in a document and does not occur in many documents, it will have a high tf-idf score.

The cosine similarity of the query vector (a bag of words, vectorized form of user input) and the document (symptoms) vector is given by the formula

$$sim(q, di) = \frac{\bar{q} \cdot \bar{di}}{\|\bar{q}\| \|\bar{di}\|}$$

where \bar{q} is the query vector and \bar{di} is the document vector

Certain optimizations in this calculation have been carried out in order to speed up the operation without losing the accuracy. $\|\bar{q}\|$ in the denominator is common across similarity computations with other documents and was ignored as it doesn't affect the ranking. $\frac{\bar{di}}{\|\bar{di}\|}$ is the normalized form of document vector and has been pre-computed and stored in the database . Since the query vector is represented as a binary vector, the simplified formula for cosine similarity of terms that are present both in query vector and the document, ie: dot product of query vector and normalized document vector will be

$$sim(q, di) = \bar{q} \cdot \bar{di} = \sum norm.tfidf$$

4.3.3 Normalized tf-idf computation

Normalized tf-idf has been computed with the help of a Python script and SQL queries.

The steps involved in the tf-idf computation are explained below.

a) Term - frequency (tf)

For each disease, symptom has been read from database using a Python script. Symptoms are then tokenized, stop words are removed and terms(words) are counted using map with term as key and value as the count. Term frequency(tf) of each term for each disease ID was then written back to database.

b) Inverse document frequency (idf)

To compute the inverse document frequency we need the total number of documents and the number of documents in which each word is occurring. This was computed with the help of below SQL queries from Python connecting to MySQL.

Total number of documents (N):

```
SELECT COUNT (disease_id) FROM DiseaseSymptoms
```

Document frequency of each term (n):

```
SELECT term, COUNT(*) AS cnt from symptoms_tfidf GROUP BY term
```

Inverse document frequency (idf) was then computed by the formula: $\log(N/n)$

c) Term frequency - Inverse document frequency (tf-idf): .This was computed by the product of above two steps and was written back to the database.

d) Normalized tf-idf

The mod of each document(symptoms for a disease) vector was computed using the below SQL query executed from Python:

```
SELECT disease_id, sqrt(sum(norm_tfidf*norm_tfidf)) FROM symptoms_tfidf  
GROUP BY disease_id
```

The above computed tf-idf of each term within each document was then divided by the mod of the document vector to do the normalization. This was then written back to the database.

After all these steps, the database table - symptoms_tfidf will have the normalized tf-idf for all the terms.

The cosine similarity between the query vector each of the document vectors has been computed by using the following query:

```
SELECT DISTINCT D.disease AS disease, D.body_location AS body_location,  
X.rank AS rank FROM disease_symptoms D JOIN ( SELECT T.disease_id,  
SUM(T.norm_tfidf) AS rank FROM (SELECT disease_id, norm_tfidf FROM  
symptoms_tfidf WHERE term IN ('fatigue','headache')) T GROUP BY  
T.disease_id ORDER BY SUM(T.norm_tfidf) desc) X ON  
X.disease_id=D.disease_id ORDER BY X.rank DESC LIMIT 10
```

4.3.4 Heart Disease Prediction

As mentioned earlier, a heart disease predictor has been incorporated into the website. This feature is used to predict if a patient is suffering from a heart disease or not based on a

definite set of heart health information. A machine learning algorithm based on randomForests has been used for training the model built on the training data set. Coding for the same has been done using R programming language.

4.3.5 Geo-coding of postal codes

To plot the location of doctors on the map, the postal code of the clinic had to be converted to latitude and longitude. This was done with the help of a Python script using 'pygeocoder' library.

This has been used in the “FindDoctor” functionality where doctors for the selected speciality are queried. Specification of the user’s locality is an optional parameter. Hence, two cases arise:

When the user locality is unspecified, all the doctors available for the given speciality are retrieved using the following query:

```
“SELECT DISTINCT dc.DoctorName, ds.Speciality, dc.ClinicName, ca.Address, cll.Latitude, cll.Lon FROM doc_speciality AS ds, doc_clinic AS dc, clinic_add AS ca, clinic_lat_long AS cll WHERE (ds.DoctorName=dc.DoctorName AND dc.ClinicName=ca.ClinicName AND cll.ClinicName=dc.ClinicName AND ds.Speciality=%s) LIMIT 15 ”
```

When the user locality is specified, the doctors for the selected speciality are retrieved based on the distance from the user’s locality with the nearest clinic on the top and the farthest at the bottom. This is done by using the geocoded pin code entered by the user and calculating the distance between every selected doctor’s clinic/hospital and the user locality . The distance in km is calculated using the Haversine formula :

```
DEGREES(ACOS(COS(RADIANS(lat1)) * COS(RADIANS(lat2)) *COS(RADIANS(long1) - RADIANS(long2)) + SIN(RADIANS(lat1)) * SIN(RADIANS(lat2))))
```

The distance in km is calculated by multiplying the result obtained in degrees with 111.045 (the distance in km for one degree).

The query used is:

```

“SELECT D.DoctorName,D.ClinicName,Latitude,Lon,D.Address,111.045*
DEGREES(ACOS(COS(RADIANS(latpoint)) * COS(RADIANS(Latitude))*
COS(RADIANS(longpoint) - RADIANS(Lon))+ SIN(RADIANS(latpoint))*
SIN(RADIANS(Latitude)))) AS distance_in_km FROM Doctors as D JOIN ((SELECT
user_locality_latpoint AS latpoint, user_locality_longpoint AS longpoint) AS p) where
D.ClinicName IN (SELECT ClinicName from doc_clinic where DoctorName in (select
DoctorName from doc_speciality where Speciality= selected_speciality)) ORDER BY
distance_in_km LIMIT 15"

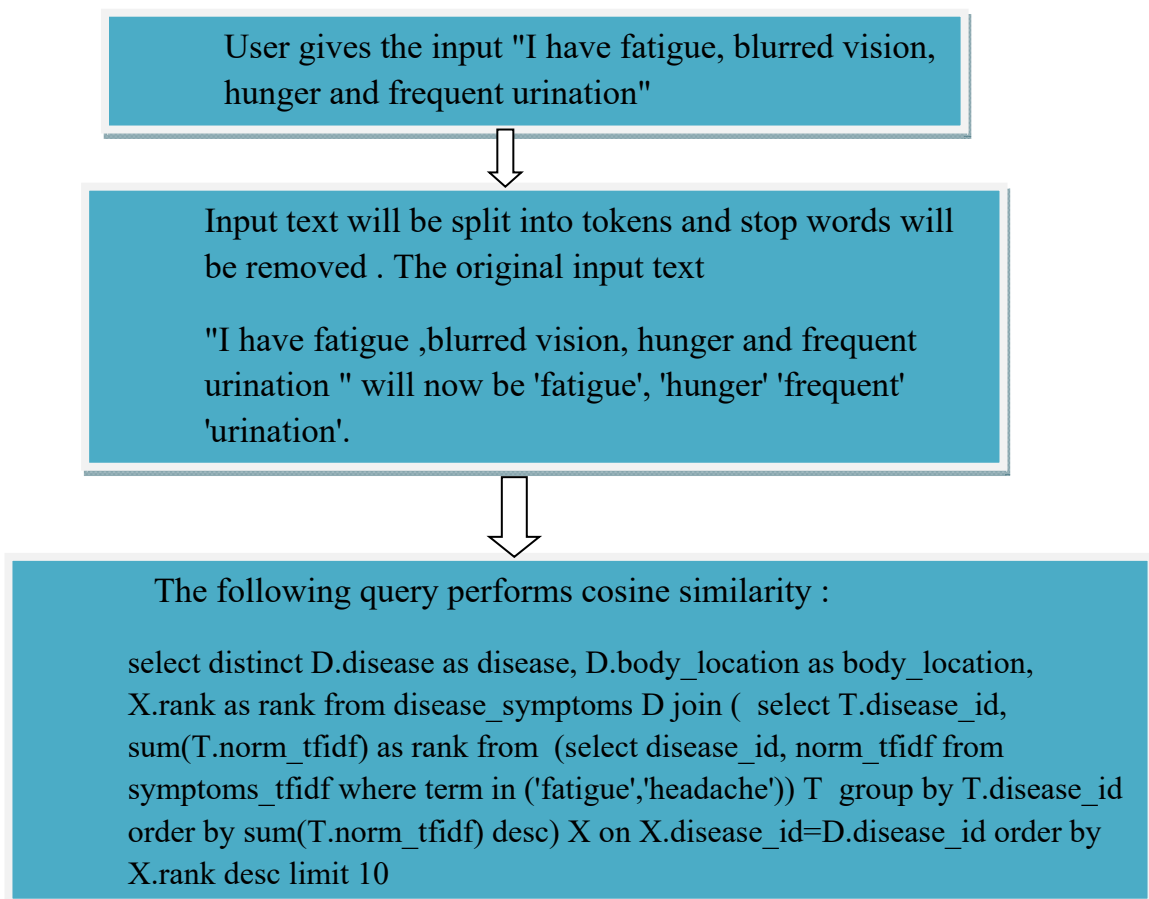
```


4.3.6 Development Environment:

The entire web application is developed in a virtual environment with Django1.10, Pip (Python Package Index) and python3 installed. Python + Django1.10 + MySQL is used for the development of this web application on the back-end. For the front-end, HTML, CSS and Materialize CSS has been used.

5. RESULTS AND DISCUSSION

The sequence of events that happen when a user inputs the symptoms and click 'Find Doctor' button are given below:





| | |
|--------------------|----------|
| Multiple Myeloma | 0.40109 |
| Bladder Diseases | 0.291377 |
| Trichomoniasis | 0.24356 |
| Diabetes Insipidus | 0.242233 |

Figure 4: Flowchart of symptom checker

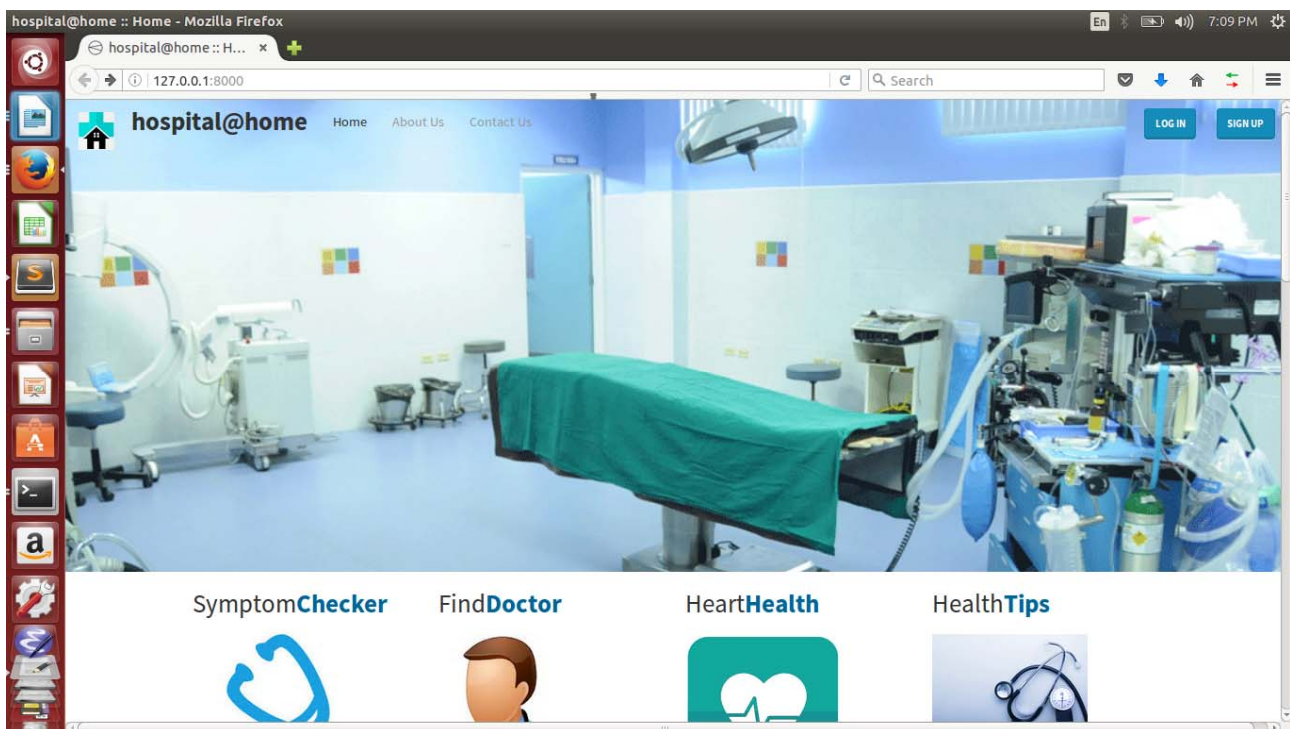


Figure 5: Front end design of the website

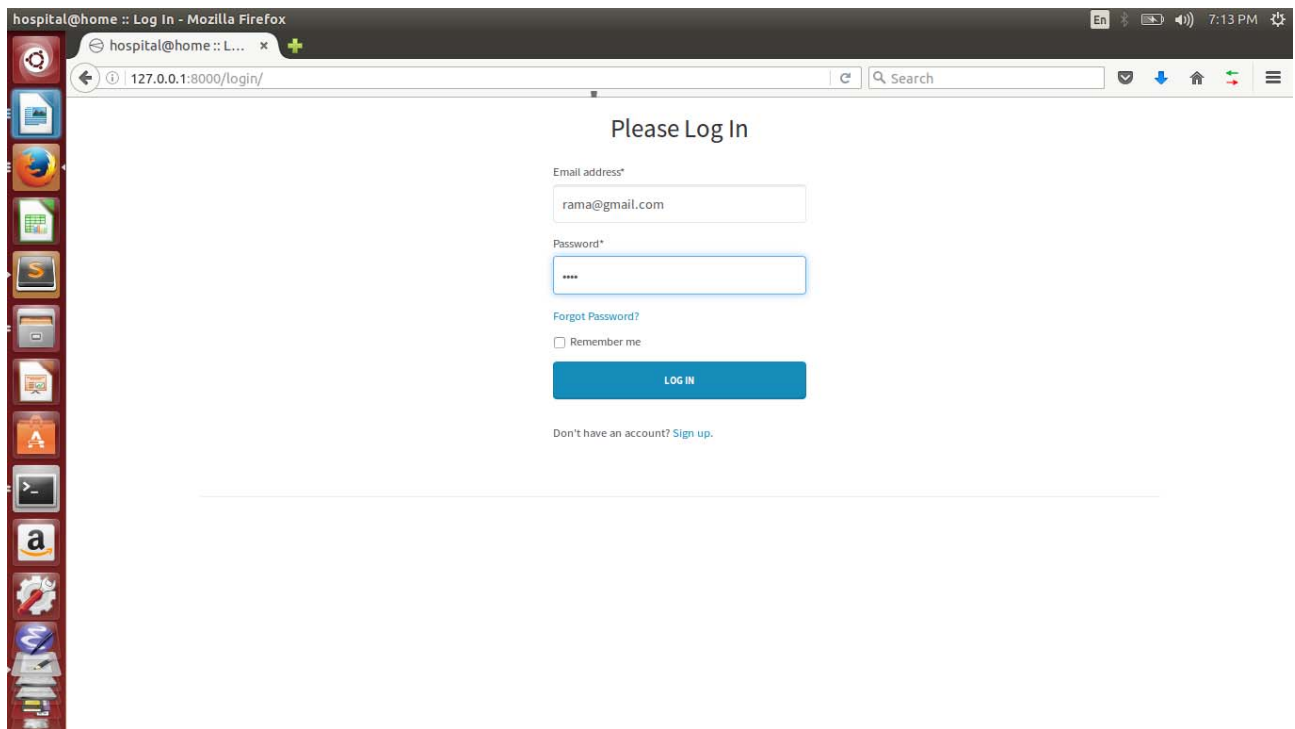


Figure 6: Login page of the website

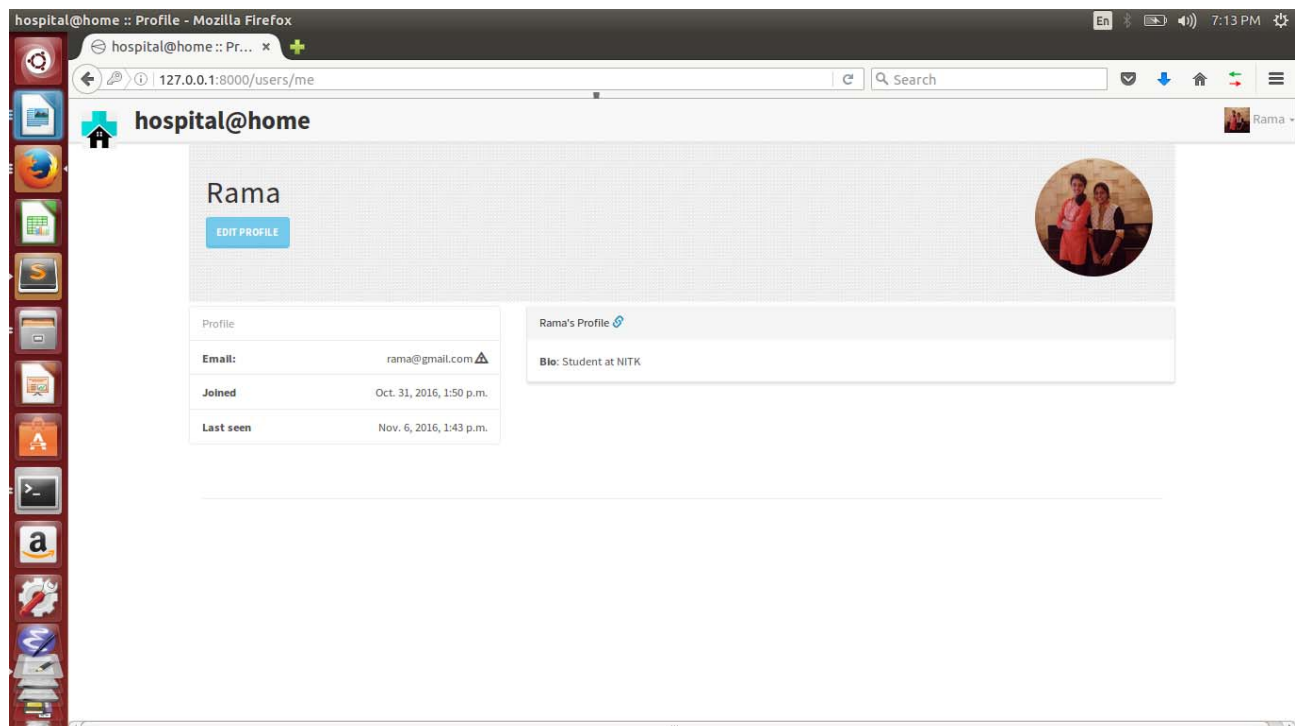


Figure 7: User profile

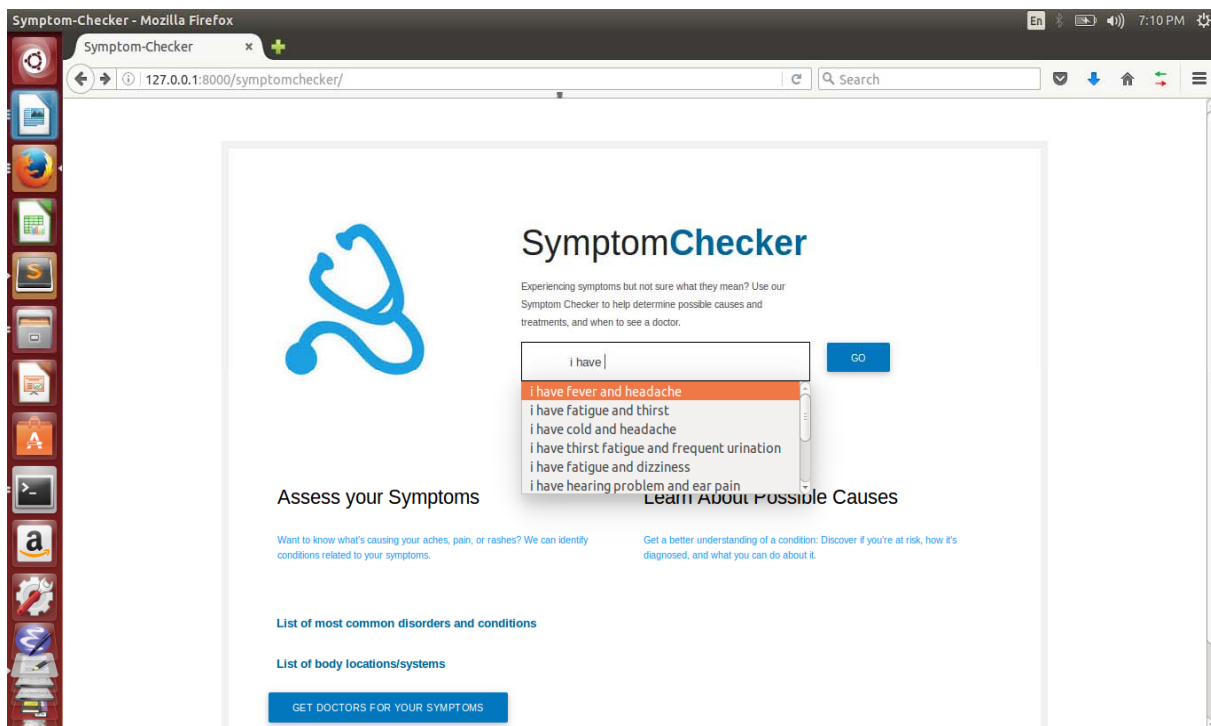


Figure 8: User enters query and checks for symptom

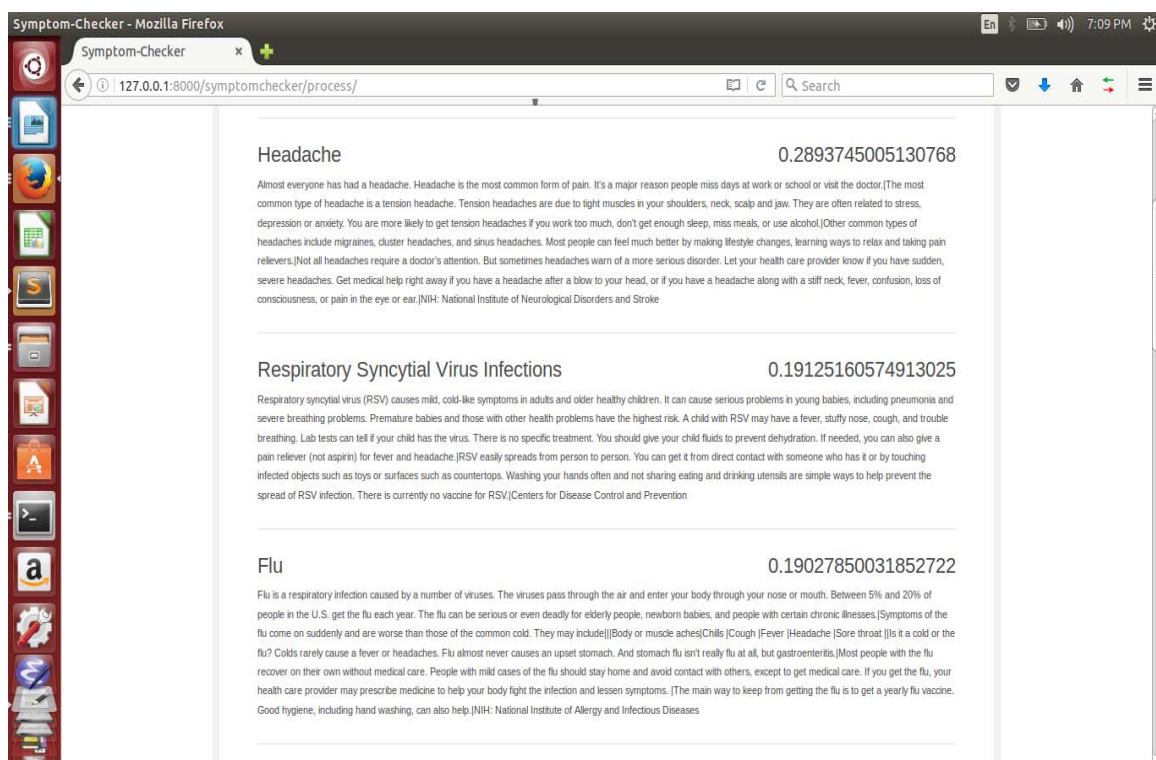


Figure 9: Result of symptom checker

After the user's health query is searched, the Symptom Checker returns all the probable diseases. It displays the name of the disease, rank of the disease and disease description. The diseases are displayed in decreasing order of the rank.

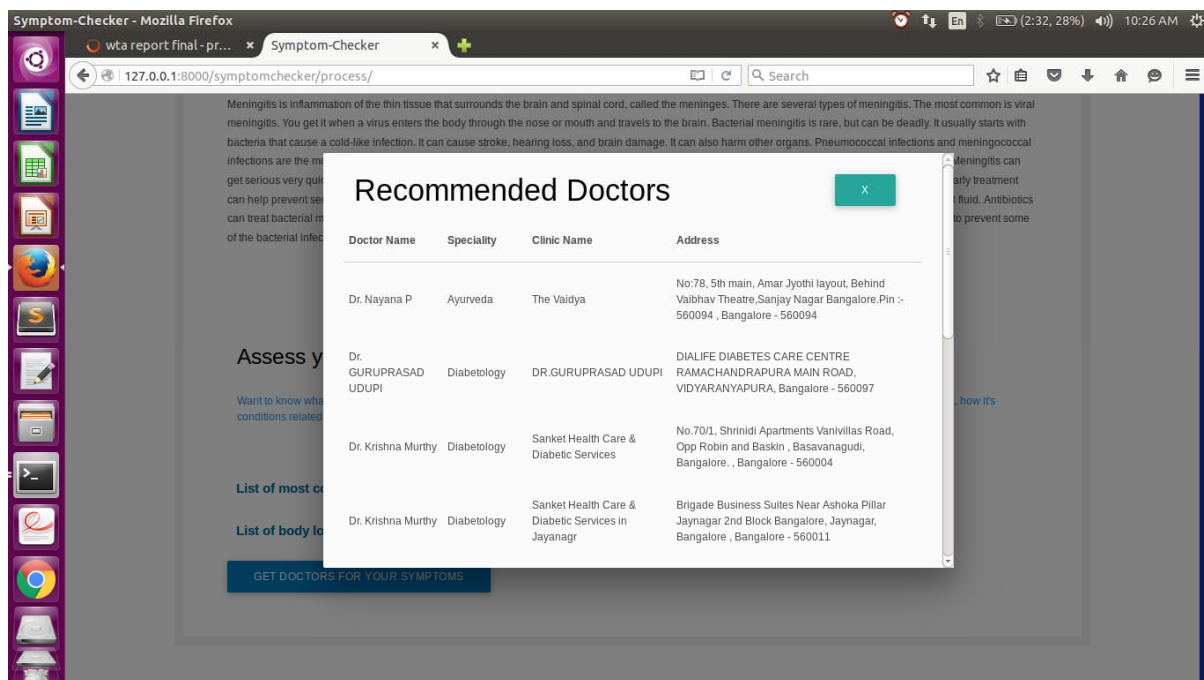


Figure 10: Get doctors for entered symptoms

The doctors available for the selected symptoms can also be checked.

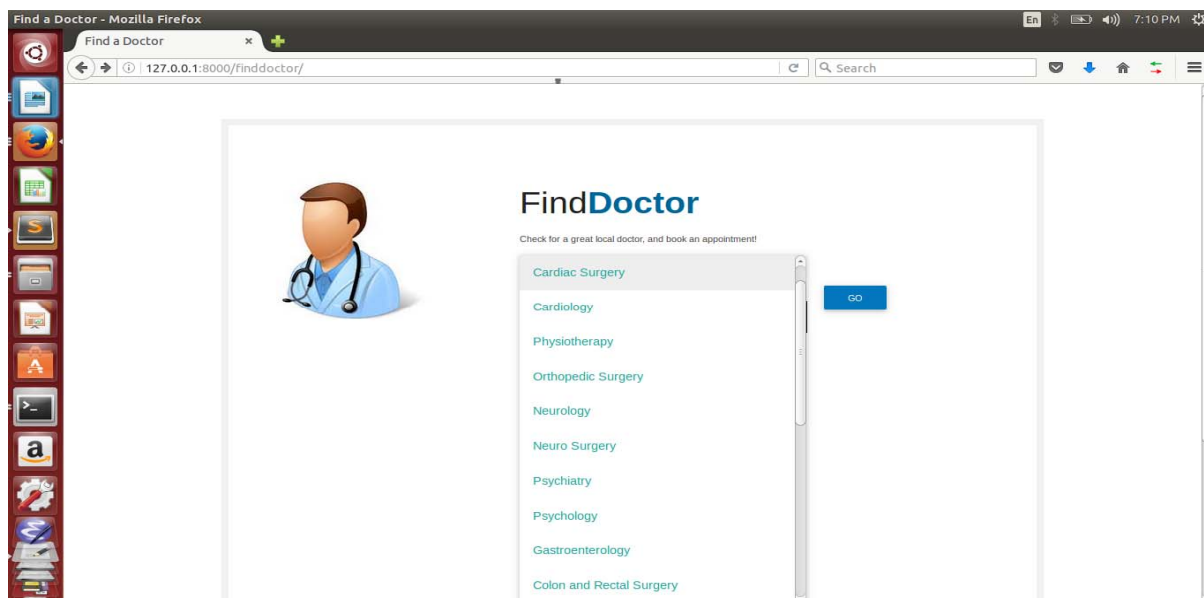
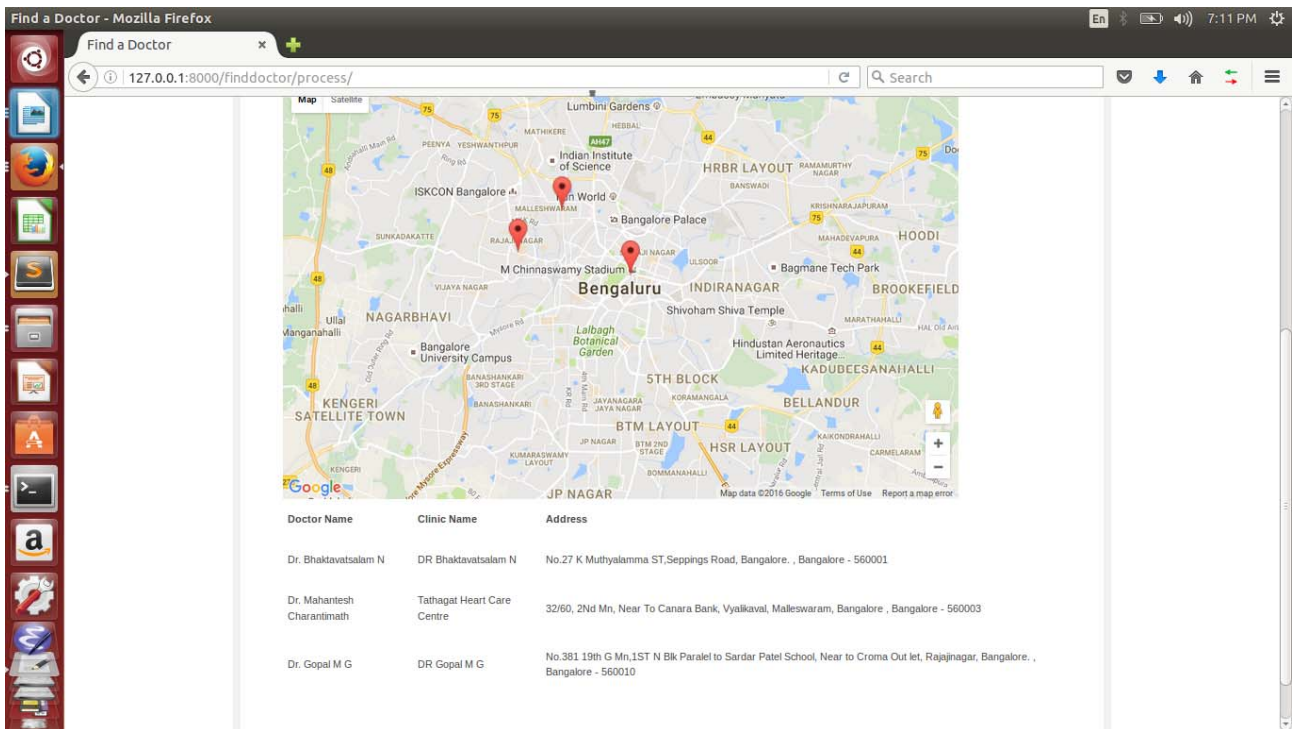


Figure 11: Find doctor with speciality

“FindDoctor” functionality allows the user to find doctors in Bangalore for the selected



speciality.

Figure 12: Mapping of doctor's location using google maps

It displays a google map where the location of the selected doctors is marked. The names of the doctors, clinic names and address is displayed below the map.

6. CONCLUSION AND FUTURE WORK

One of the major challenges faced during this project was to collect the right data. Initially, plans to use anonymized patient record data for building our application existed. However, accurate medical data is generally difficult to obtain due to its sensitive nature and a suitable dataset could not be obtained.

Due to these challenges, the text mining approaches were used.

6.1 Limitations

One of the main limitations of the web application is the lack of “personalization” depending on patient medical history. For example, if a diabetic patient is suffering from pain and swelling in the foot, it is more likely to be a “diabetic foot” disease rather than an orthopaedic injury. Future versions of this application may be built to include this feature, conditional to availability of real medical data that allows us to draw this kind of relationships.

6.2 Further Scope

We have identified two ways in which this application could be further extended.

- I. Feedback and Review system.
- II. Hospital Booking automation

Feedback and Review system: A future version of this app may include a 'Feedback' section for people to review doctors and medical establishments. Social network elements may also be included to digitize the large "word-of-mouth" review system existent in every community.

Hospital booking automation: In many hospitals, patients register themselves at the registration desk by describing their symptoms to the nurse. The patients are then redirected to the doctors based on the description of the symptoms. This "Expert System" model is, however, limited by the lack of scalability and suffers from human errors, higher cost and more time spent. The "FindDoctor" functionality can be automated by acting as the first interface between the patient and the hospital.

6. REFERENCES

- Hlaudi Daniel Masethe, Mosima Anna Masethe , (2014) "Prediction Of Heart Disease Using Classification Algorithms" Proceedings of the World Congress on Engineering and Computer Science 2014 Vol II
- Faisal Rahutomo*, Teruaki Kitasuka, and Masayoshi Aritsugi "Semantic Cosine Similarity"
- Machine Learning :Text feature extraction (tfidf)-I
<http://blog.christianperone.com/2011/09/machine-learning-text-feature-extraction-tf-idf-part-i/>
- Machine Learning :Text feature extraction (tfidf)-II
<http://blog.christianperone.com/2011/09/machine-learning-text-feature-extraction-tf-idf-part-ii/>
- Django documentation <https://docs.djangoproject.com/en/1.10/>

- R Random Forest Algorithm https://www.tutorialspoint.com/r/r_random_forest.html