SRS DOCUMENT

TWO SEMESTER PROJECT

OCTOBER 21, 2022

KCA UNIVERSITY
KCA 00101 NAIROBI

NAME: SANTOS OMONDI OKELLO

COURSE: Bsc.IT

REG: 19/05036

SUP: MR COLLINS ONDIEK

DISEASE PREDICTION SYSTEM

TWO SEMESTER PROJECT

AUTHOR NOTE

Data collection and also preliminary research and analysis was conducted and also sponsored by the school of health and science in JKUAT and Facaulty of computing and information technology in KCA University respectively. The portion of these information and finding was presented as a poster in 2022 online in Virtual campus KCA University supervised by Mr Collins Ondiek. Response action concerning this article should be addressed to Dean of school of technology Dr Mwendia, Madam Merab Omondi, and also the support and managing staff in KCA University.

First M. Last www.santostimelapse.

ABSTRACT

“Disease Prediction” system based on predictive modeling predicts the disease of the user on the basis of the symptoms that user provides as an input to the system. The system analyzes the symptoms provided by the user as input and gives the probability of the disease as an output Disease Prediction is done by implementing

the Naïve Bayes Classifier. Naïve Bayes Classifier calculates the probability of the disease. Therefore, average prediction accuracy probability 60% is obtained.

PURPOSE

The primary goal is to develop a prediction engine which will allow the users to check whether they have diseases like malaria, tuberculosis, typhoid, diabetes or heart disease et cetera sitting at home when feeling sick. The user don't need visit the doctor unless he or she has a strong disease that required physical checkup, for further treatment. The prediction engine requires a large dataset and efficient machine learning algorithms to predict the presence of the disease. Pre-processing the dataset to train the machine learning models, removing redundant, null, or invalid data for optimal performance of the prediction engine

DOCUMENT CONVENTIONS

This document uses the following conventions.

DB	Database
DDB	Distributed Database
ER	Entity relationship

INTENDED AUDIENCE AND READING SUGGESTIONS

We live in a disease world, why can't patient find a clear way to help themselves. The General disease prediction system predicts chances of presence of a disease present in a patient, on the basis of our system, it will also recommend necessary precautionary also measures required to treat the predicted disease.

PROJECT SCOPE

Disease prediction has a potential to benefit stakeholders such as the gorvenment and health insurance. It can identify patients at risk of disease or health condition and most importantly, the system also stores patients data for previous disease outcome because we believe a disease can make way of another one if not treated to perfection.

System Overview

Infrastructure of disease consists of five parts: Wearable, fixed devices and cloud server, mobile application and website.

Wearable device consists of various sensors that constantly gather data from the user. The device will join all the data coming from sensors and push them to fixed device for further computation.

- Fixed device is responsible for processing the data in real time. It is the main computation unit of the system. Fixed device also maintains the communication with the cloud server, and sends user data in a daily basis.
- Cloud server is the main storage unit for past user data. It acts as a bridge between the wearable device and mobile and web applications. It will handle users and will be answering requests coming from mobile application and website.
- Mobile application will be used to visualize and show daily user data. The app will be an easy to use platform suited for end users.

Website will be used by users to see more detailed data than the mobile application. The main purpose of the website will be analysis of this data. It will be more suited for data analysts

OVERALL DESCRIPTION

A distributed disease prediction system stores the following information.

- DP:

  It includes the database that holds symptoms and necessary precaution to be takes.

- Chat bot:

  It includes a process built with restored AI with naïve bayes algorithm to be able to detect disease and be able to provide local or prescribed drug as a AID.

- Patient description:

In order to predict disease several factors has been consider such as body mass index, cholesterol level, blood sugar, blood pressure and so on

Naïve BAYES

Algorithm implemented

The algorithm implemented in this project is Naïve Bayes Classifier

Naïve Bayes classifier depends on Bayes Theorem

Equation 1:

$$P(Y|X1, \dots \dots \dots . . , Xn) = \frac{P(Y)P(X1, \dots \dots , Xn|Y)}{P(X1, \dots \dots . Xn)}$$

Where,

Y is the class variable

X1, , X2, … … . . , Xn are the dependent features

From equation 1 we get equation 2 as:

P(Disease|symptom1, symptom2, … … . , symptomn)

$$\frac{\mathrm{P(Disease)P(symptom1, \dots\ \dots\ .., symptomn|Disease)}}{\mathrm{P(symptom1, symptom2, \dots\ \dots\ \dots\ \dots\ ..Symptomn)}}$$

Using the naive independence assumption :

P(symptom1, … … . . , symptom|Disease) = P(Symptomi |Disease)

Where i= 1, 2, …………,n

Equation 3:

P(Disease|symptom1, symptom2, … … . , symptomn)

$$\frac{\mathrm{P(Disease)P(Symptomi\ |Disease)}}{\mathrm{P(symptom1, symptom2, \dots\ \dots\ \dots\ \dots\ ..Symptomn)}}$$

So the relation becomes:

Equation 4:

P(Disease|symptom1, symptom2, … … . , symptomn)

$$\frac{\mathrm{P(Disease)} \prod \mathrm{P(Symptomi\ |Disease)}}{\mathrm{P(Disease)} \prod \mathrm{P(Symptomi\ |Disease)\ n\ i = 1\ P(symptom1, symptom2, \dots\ \dots\ \dots\ \dots\ ..Symptomn)}}$$

Since P(symptom1, symptom2, … … … … . .Symptomn) is constant, we can use the

following classification rule:

P(Disease|symptom1, symptom2, … … . , symptomn)

= P(Disease)∏P(Symptomi |Disease)

P(Disease|symptom1, symptom2, … … . , symptomn)∝

P(Disease) ∏ P(symptomi|Disease)

ˆY= ARG MAX P(Disease) ∏P(Symptomi |Disease)

The value P(Symptomi |Disease) of can be calculated by using multinomial Naïve Bayes which is given by:

$(symptomi\ |Disease) = \frac{\mathrm{Nyi} + \alpha}{\mathrm{Ny} + \alpha�}$

Where:

Nyi= Frequency of same disease in the dataset

Ny= Total symptoms of the particular disease

n= total symptoms in the dataset

α=1, known as Laplace Smoothing

The value of P(Disease) can be calculated by using Laplace Law of Succession which is given by:

$\mathrm{P(Disease)} = \frac{\mathrm{N(Disease)} + 1}{\mathrm{N} + 2}$

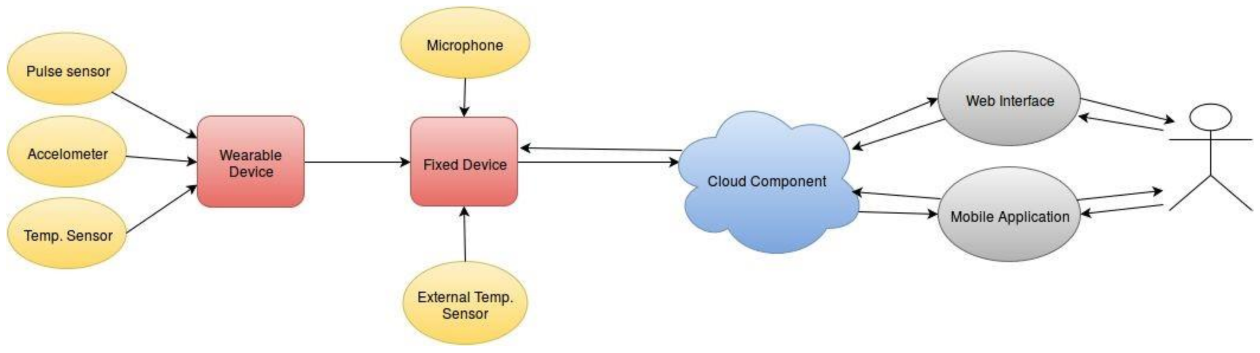
Where,

N (Disease) = Frequency of the same disease in the dataset

N= Total disease in the dataset

Product functions

In this project, sensors placed on wearable device will collect information from person with a configured frequency and transmit this data to fixed device. This fixed device will also have sensors collecting data and will process all sensor data in real time. Unless this device detects an emergency situation, it will send processed data to cloud component in a daily basis. If it detects an emergency such as heart attack, predetermined people will be alerted via an automatic phone call or SMS etc.. The cloud component will take data from fixed device and apply machine learning algorithms to add a meaning to data. This component will understand light stages of sleep in order to wake up the user in the right time, will figure out personalized sleep patterns and hopefully will diagnose some health problems like epilepsy crisis or sleep apnea. Users will be informed about their sleep based on time and sensor choice through web application and mobile application



PRODUCT FEATURES

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

Data Model and Description

Class diagrams are as follows:

Fixed Component Class Diagram

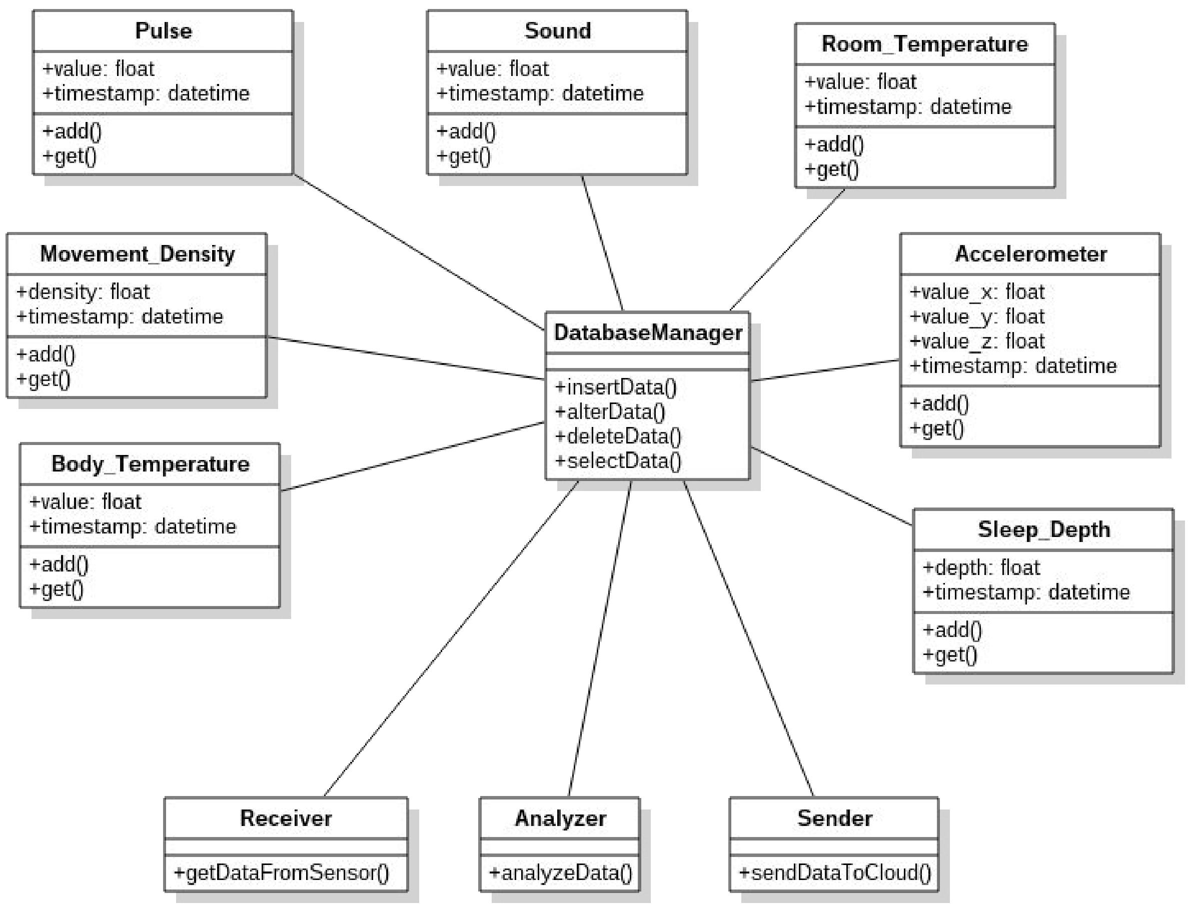


Figure 1 user details that should be available

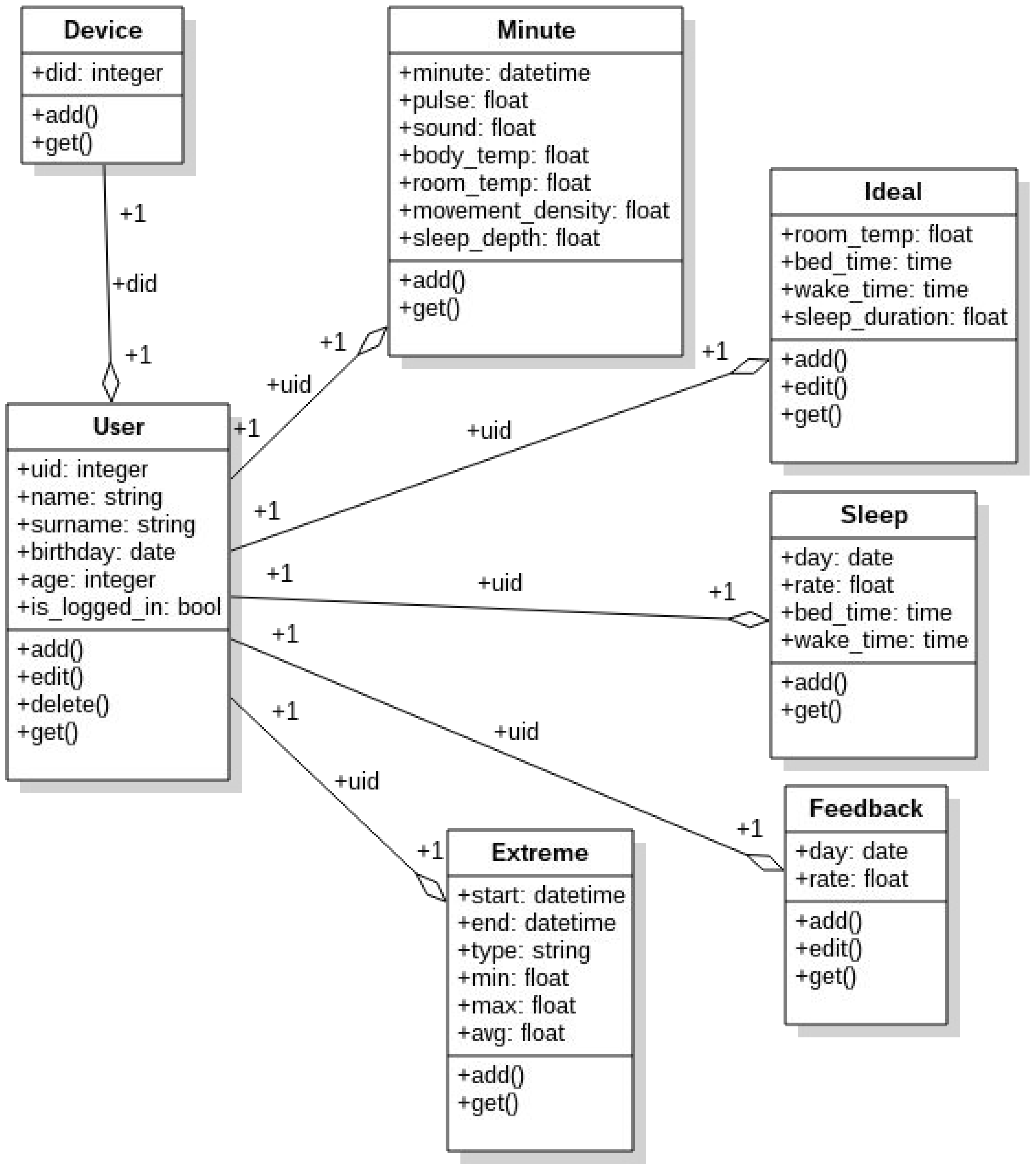
Class	**Description**

DatabaseManager	It is responsible for data transactions **Methods:** ● **insertData:** inserts new sensor data into database ● **alterData:** alters a sensor data from database ● **deleteData:** deletes a sensor data from database ● **selectData:** returns a sensor data from database
Receiver	It is responsible for getting the sensor data **Methods:** ● **getDataFromSensor:** gets data from a sensor
Analyzer	It is responsible for analyzing the sensor data **Methods:** ● **analyzeData:** analyzes the sensor data
Sender	It is responsible for sending sensor data to cloud **Methods:** ● **sendDataToCloud:** sends the sensor data to cloud
Pulse	It represents the pulse values coming from pulse sensor **Attributes:** ● **value:** pulse value ● **timestamp:** the time of pulse value **Methods:** ● **add:** adds a new pulse value with timestamp ● **get:** returns a pulse value

Sound	It represents the digitalized sound values coming from microphone **Attributes:** ● **value:** sound value ● **timestamp:** the time of sound value **Methods:** ● **add:** adds a new sound value with timestamp ● **get:** returns a sound value

Room_Temperature	It represents the room temperature values coming from thermometer **Attributes:** ● **value:** room temperature value ● **timestamp:** the time of temperature value **Methods:** ● **add:** adds a new temperature value with timestamp ● **get:** returns a temperature value
Body_Temperature and mass	It represents the body temperature values coming from thermometer **Attributes:** ● **value:** body temperature and mass value ● **timestamp:** the time of temperature and mass value **Methods:** ● **add:** adds a new temperature and mass value with timestamp ● **get:** returns a temperature and mass value

Accelerometer	It represents the x,y,z acceleration values coming from accelerometer sensor **Attributes:** ● **value_x:** x axis acceleration value ● **value_y:** y axis acceleration value ● **value_z:** z axis acceleration value ● **timestamp:** the time of acceleration values **Methods:** ● **add:** adds a new accelerometer value with timestamp ● **get:** returns a accelerometer value
Movement_Density	It represents the movement density values calculated using accelerometer values of each timestamp **Attributes:** ● **density:** density of movement at timestamp ● **timestamp:** the time **Methods:** ● **add:** adds a new density value ● **get:** returns a density value
Sleep_Depth	It represents the sleep depth values calculated using all sensor values of each timestamp **Attributes:** ● **density:** density of movement at timestamp ● **timestamp:** the time **Methods:** ● **add:** adds a new density value ● **get:** returns a density value



Class	**Description**

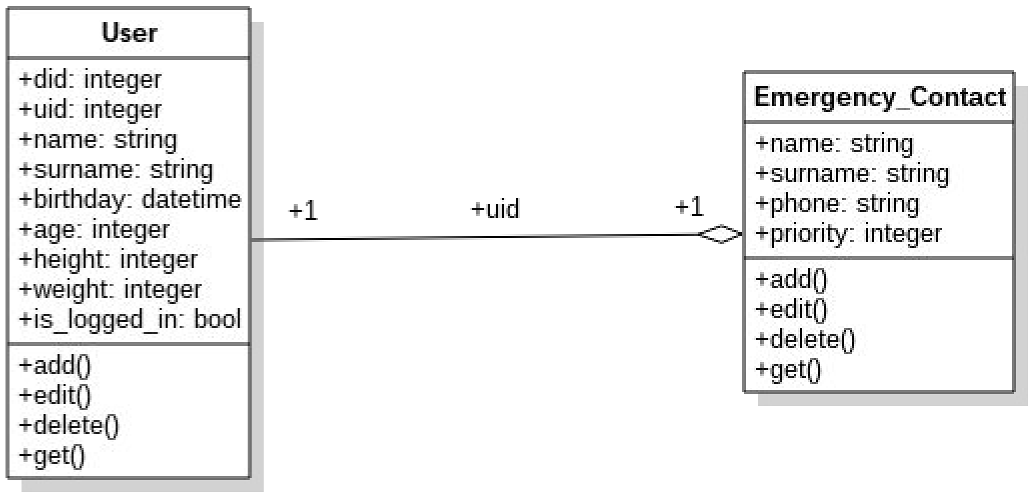
Device	It represents the fixed device ids which are in use **Attributes:** ● **did:** device id **Methods:** ● **add:** adds a new device id
Minute	It represents the average values of all sensors at 1 minute intervals. **Attributes:** ● **uid:** id of the user ● **minute:** time of average sensor values ● **pulse:** pulse sensor value ● **sound:** sensor value ● **body_temp:** body temperature value ● **body_mass:** body temperature value ● **room_temp:** room temperature value ● **movement_density:** density of acceleration values ● **sleep_depth:** sleep depth score of user **Methods:** ● **add:** adds new minute values ● **get:** gets minute values

Ideal	It represents the ideal sleeping environment for a user **Attributes:** ● **uid:** id of the user ● **room_temp:** room temperature value ● **bed_time:** time to go to bed ● **wake_time:** time to wake up ● **sleep_duration:** sleep duration **Methods:** ● **add:** adds new ideal values ● **edit:** edits the ideal values ● **get:** gets ideal values

User	It represents the user **Attributes:** ● **uid:** id of the user ● **name:** name of the user ● **surname:** surname of the user ● **birthday:** birthday of the user ● **age:** age of the user ● **is_logged_in:** value states whether the user is logged in **Methods:** ● **add:** adds user ● **edit:** edits the user ● **delete:** deletes the user ● **get:** gets the user

Sleep	It represents a sleep of a user **Attributes:** ● **uid:** id of the user ● **day:** day of sleep ● **rate:** calculated sleep rate ● **bed_time:** the time user went to bed ● **wake_time:** the time user woke up **Methods:** ● **add:** adds new sleep ● **get:** gets the sleep
Extreme	It represents the extreme cases that sensors values got **Attributes:** ● **uid:** id of the user ● **start:** start time of the extreme case ● **end:** end time of the extreme case ● **type:** sensor name ● **min:** minimum value the sensor got ● **max:** maximum value the sensor got ● **avg:** average value the sensor got **Methods:** ● **add:** adds an extreme case ● **get:** get an extreme case

Feedback	It represents the feedback of the user about their sleep **Attributes:** ● **uid:** id of the user ● **day:** day of sleep ● **rate:** rate of user to their sleep **Methods:** ● **add:** adds a feedback ● **edit:** edits a feedback ● **get:** gets a feedback



Class	**Description**
User	It represents the fixed device ids which are in use **Attributes:** ● **did:** device id ● **uid:** user id ● **name:** name of the user ● **surname:** surname of the usee ● **birthday:** birthday of the user ● **age:** age of the user ● **height:** height if the user ● **weight:** weight of the user ● **is_logged_in:** value states whether the user is logged in **Methods:** ● **add:** adds a new user ● **edit:** edits the user ● **delete:** deletes the user ● **get:** gets the user

Emergency_Contact	It represents the emergency contact person **Attributes:** ● **uid:** id of the user ● **name:** name of the contact person ● **surname:** surname of the contact person ● **phone:** emergency contact number ● **priority:** priority value of the contact person **Methods:** ● **add:** adds a new emergency contact ● **edit:** edits the emergency contact ● **delete:** deletes the emergency contact ● **get:** gets the emergency contact

USER CLASS AND CHARACTERISTICS

The system will use NAÏVE BAYES algorithm to enable and moniterize the system. The main impact of these system is to help predict disease that patients are going through and provide a clear solution to cure the disease. The solution can be .1 use the prescribed drug by the system after detecting the disease .2 use Local drug if the modern drug is not available at the moment or visit a hospital immediately

Example.

Local way to help stomach ache is by a patient putting hot towel on his/her abdomen. The heat can relax the muscle and relieve cramping or taking a hot shower may also help, secondly, is by using prescribed drug like loperamide to help reduce the pain. When it continue you are advised to visit hospital immediately

The Patient should be able to do the following functions:

Make doctor request

Comment section

Search disease cure

Interact with the chatbot to get help

User Use Cases

Use Case	**Description**
Login	Users need to login to start using system.
Register	Users need to register themselves and their devices to system.
Check profile	Check disease profile of previous prediction
View Profile	Users can see their profile which includes health information.
Edit Profile	Users can change their personal & health information.
View Emergency Records	Users can see history of emergency situations.
View Disease Predictions	Users can see possible diseases predicted by Naïve Bayes algorithm using the system.
Naïve	Users can see their information filtered by time and sensor choice.
View Dashboard	View the recent predicted disease by chatbot .
View Analytics	Users can see the results of the applied machine learning algorithms.

Edit emergency contacts	Users can edit contacts to be informed in emergency situations

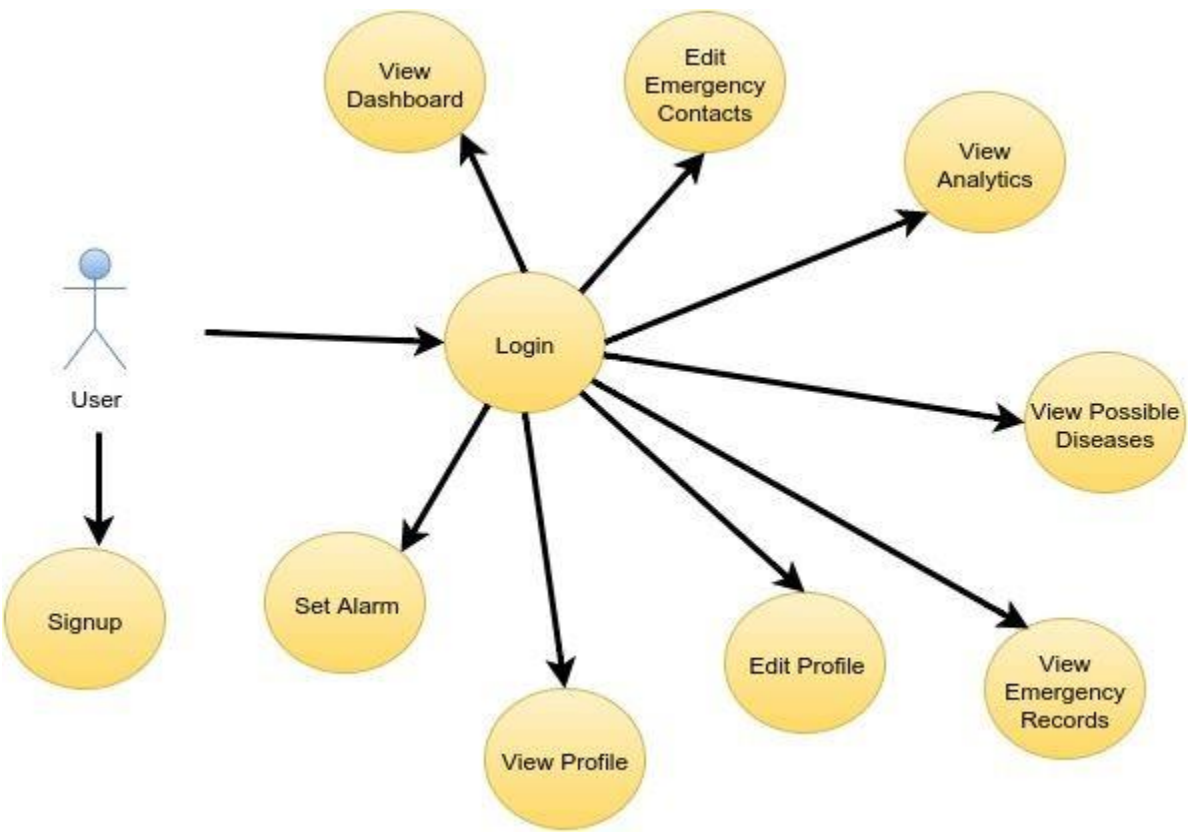


Figure 2

Actor survey

User

This actor refers to the people who own a device. A user will be able to view his/her personal information based on time and sensor choice. Users also see health record, analysis and configuration of devices.

Emergency Contact

This actor refers to the people to be informed by system when there is an extreme situation.

User Interfaces

User interface of web application will be comprehensible and easy to use. There will be 4 basic pages; dashboard, analytics, event history, disease prediction. The characteristics of web application's user interface will be as follows: Interface will be shown by web browsers. There will be dashboard page. In dashboard page, user will be able to see charts showing each sensor data separately and an initial chart that includes all sensor data. In analytics page, user will be able to see his or her results from application of machine learning algorithms. On event history page, user will be able to view list of past events detected by the device. On disease predictions page, user will be informed about the diseases that s/he might have.

System will also have an android application. Application will provide same functionality as the web application, but it will have a simpler interface. In addition, mobile application will show the user's own profile information in a daily basis

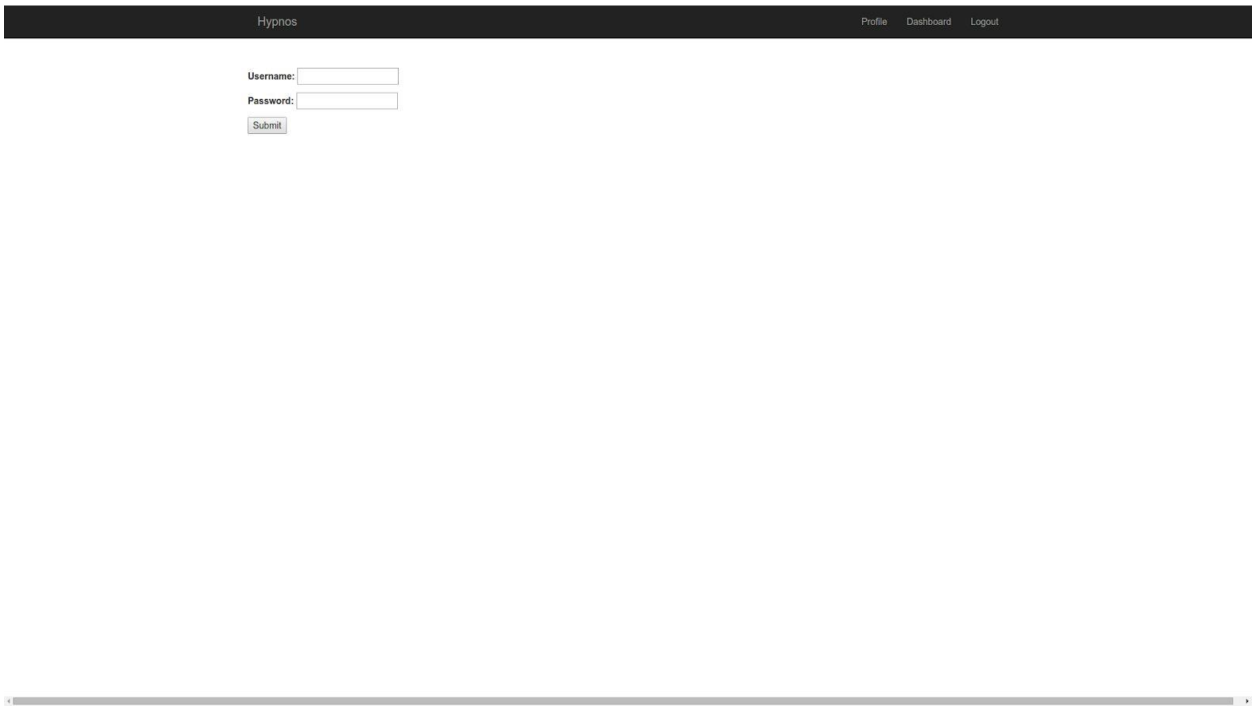


Figure 3 Sign in

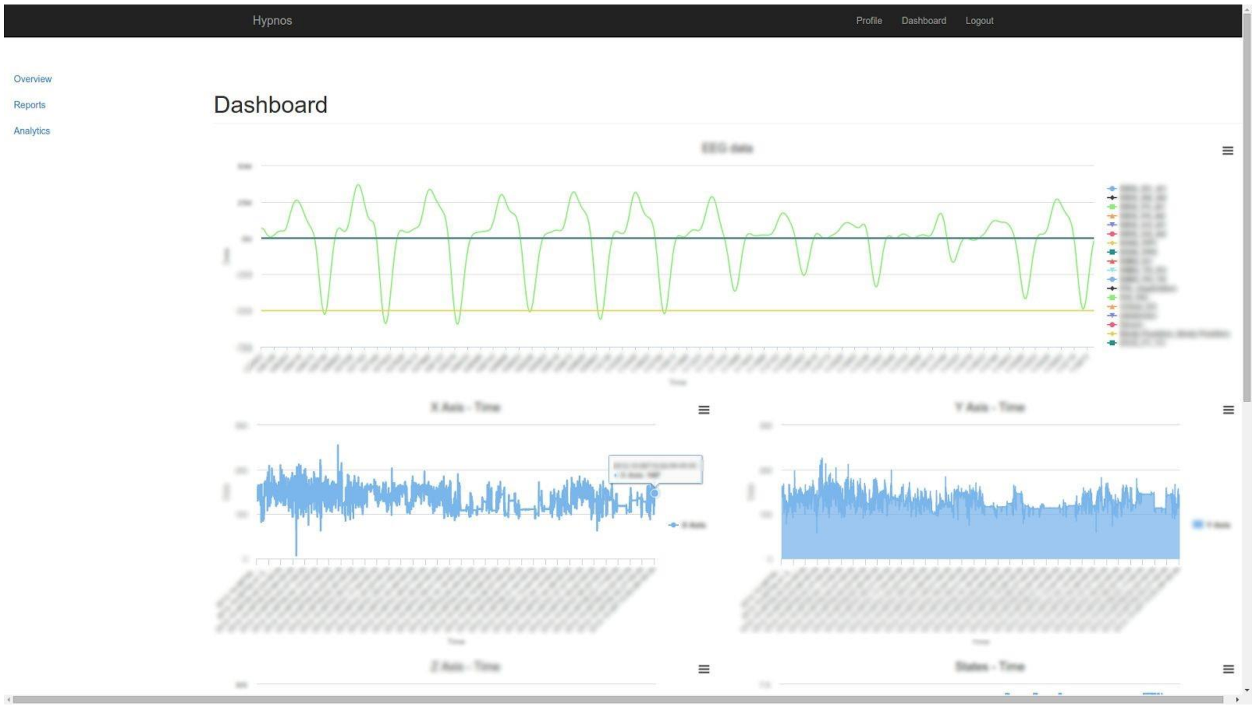


Figure 4 dashboard and recent patient data

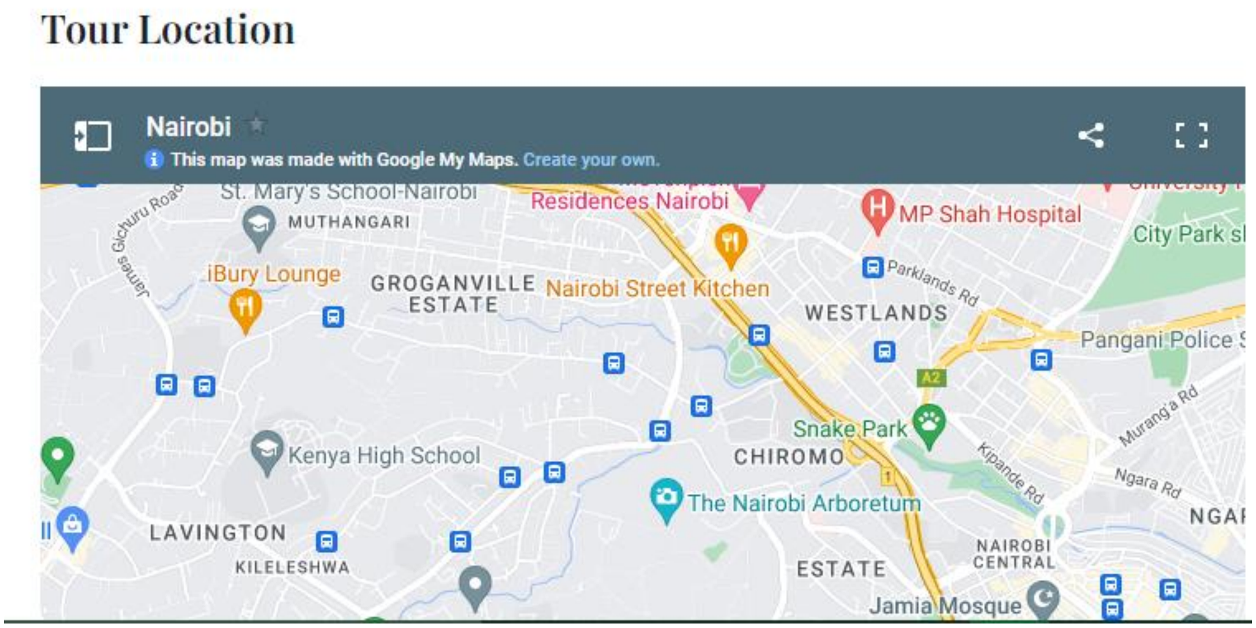
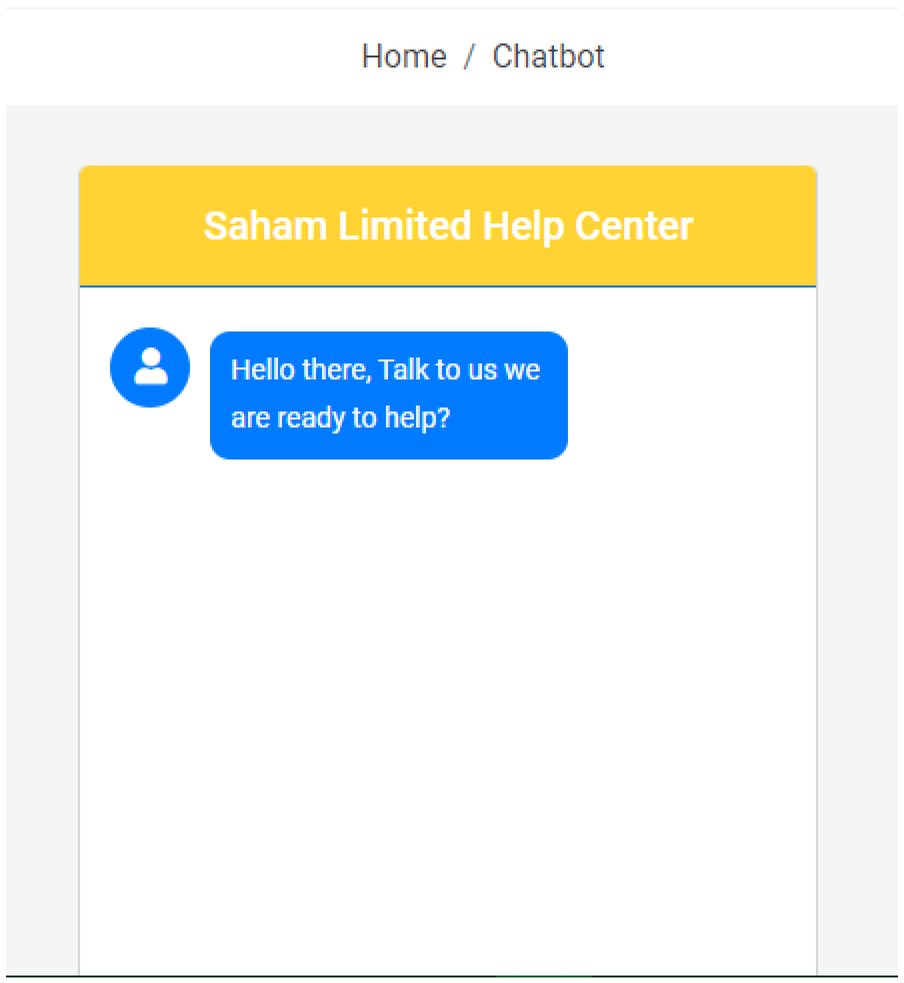


Figure 5 geographical location for emergency


Home / Chatbot

Saham Limited Help Center

Hello there, Talk to us we are ready to help?

Figure 6 chatbot to help predict disease

OPERATING ENVIRONMENT

Operating environment for the disease prediction system is as listed below

Distributed database

Client/server system

Operating system: windows

Database: sql+ database (mysql)

Platform: XAMPP

DESIGN AND IMPLEMENTATION CONSTRAINTS

The global schema, fragmentation schema, and allocation schema.

SQL commands for the above queries/application

Implement the database at least using a centralized database management system

ASSUMPTION DEPENDENCIES

Let us assume that this is a distributed disease prediction system and it is used in the following application:

Diseases that can be predicted using machine learning are simple cart, naïve Bayes, svm and random forest are used for prediction and analyze the diabetes data

The secondary aim is to develop a web application that allows users to predict heart disease, malaria, tuberculosis, diabetes et cetera utilizing the prediction engine

To implement the IT in real world problems.

To help general practice doctors, nurses, nursing students and to assist the eye patients as first aid diagnosis

SYSTEM FEATURES

DESCRIPTION AND PRIORITY

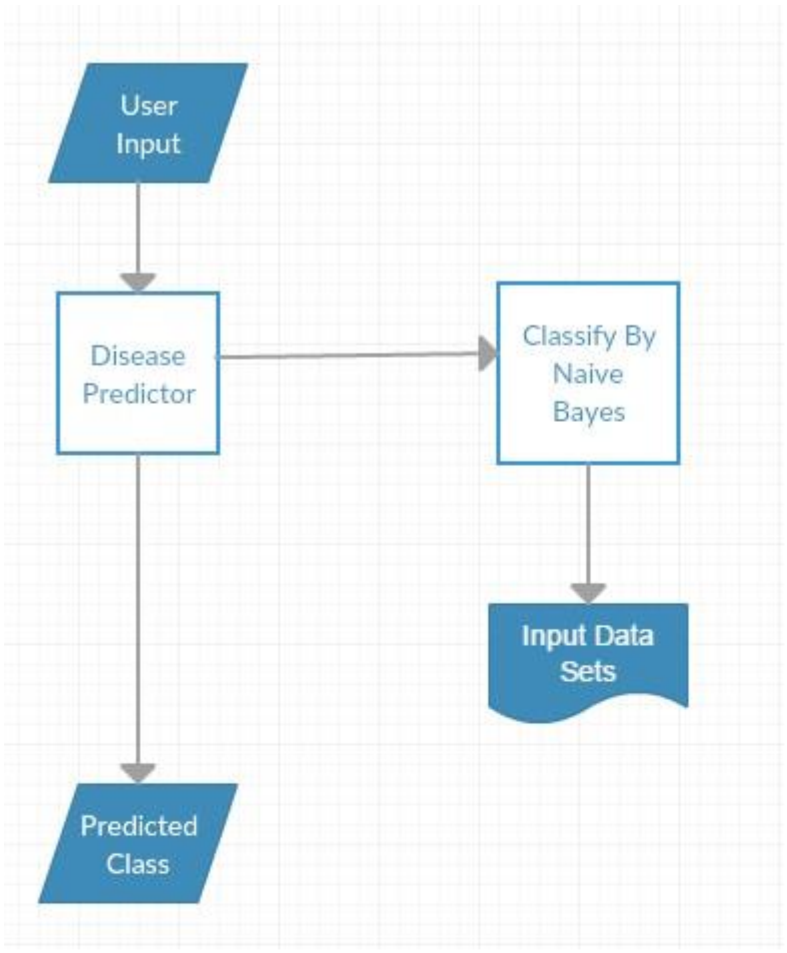
The disease diagnosis system will permit end-users to predict disease like malaria, tuberculosis, typhoid, heart disease et cetera

STIMULUS/RESPONSE SECQUENCES

IMPLEMENTATION

Disease Predictor is the ability to predict the disease that has been provided to the system.

For disease prediction, we need to implement the naïve Byes Classifier



As shown in the figure the input data sets are classified using Naïve Bayes classifier. The sample input data sets is shown below

- Sample Data Sets

Symptoms	Disease
Runny nose ,Sore throat ,Cough ,Congestion, body aches, headache ,Sneezing , fever	Common cold
Fever ,profuse sweating ,headache ,nausea ,vomiting ,diarrhea ,anemia ,muscle pain ,convulsions ,coma bloody stools ,shaking chill	Malaria

poor appetite ,abdominal pain ,headaches ,generalized aches and pains ,fever ,lethargy ,intestinal bleeding or perforation ,diarrhea , constipation	Typhoid

Naïve Bayes classifier uses the following rule to classify the datasets:

$$Y = \text{ARG MAX } P(\text{Disease}) \prod_{i=1}^{n} P(\text{symptomi}|\text{Disease})$$

User gives input to the system. The input consists of symptoms. The user marks the symptoms due to which the user is feeling unwell

☐ 1. Fever
☐ 2. Cough
☐ 3. Vomiting

The “Disease Predictor” system predicts the disease according to the input data sets and calculates the probability of the disease.

The sample output is given as:

Sample Output

Disease Name	Probability
Typhoid	0.5%
Malaria	0.3%
Flu	0.333%

FUNCTIONAL RECQUIREMENTS

Other system features include:

DISTRIBUTED DATABASE

Distributed database implies that a single application should be able to operate transparently on data that is spread across a variety of different databases and connected by a communication network as network

CLIENT/SERVER SYSTEM

The term client/server refers primarily to an architecture or logical division of responsibilities, the client is the application (also known as front-end), and the server is the DBMS (also known as the back-end).

A client/server system is a distributed system in which,

Some sites are client sites and others are server sites.

All the data resides at the server sites.

All application execute at the client sites.

EXTERNAL INTERFACE REQUIREMENTS/BUDGET

HARDWARE INTERFACES

hp laptop, mouse, Themometer, DEXA et cetera

SOFTWARE INTERFACES

Following are the software used for the disease prediction system

	Software Product	**Source**
Client on internet	Web Browser	SE
Cloud OS	Ubuntu 14.04, window 10 pro	http://www.ubuntu.com/
Web Server	Django 1.9	https://www.djangoproject.co m/
Database Server	PostgreSQL 9.4 , mysql	http://www.postgresql.org/
Development End	HTML5, JS, Bootstrap, Android, MPAndroidChart, Highcharts	Latest & Stable Version
Machine Learning	Python, Scikit, Numpy, Scipy, Matplot	Latest & Stable Version

Communications Interfaces

Various communication protocols will be used for different parts of the system.

Communication between fixed and wearable device will be done with Zigbee protocol.

All sensor data in wearable device will be concatenated and sent as a string A + ”-” + T + “-” P + “-” (A: Accelerometer, T: Temp, P: Pulse) to the fixed device via Xbee. Fixed device will parse the string and get the sensor values as integer values.

Communication between fixed device and cloud server will be done with JSON. Fixed device will send the data through Wi-Fi. TCP/IP standards will be used for packet transmission as personal data should be kept securely.

JSON data will consist of sensor values retrieved from wearable device, sensor values of the temperature and sound sensors on the fixed device and timestamp of when the sensor values were retrieved. Cloud will parse this JSON upon retrieving, and save it to the database with corresponding user ID.

Mobile applications and website will be sending requests to cloud. Firstly, requested data will be converted to JSON. Then,

- JSON data will be fed to dynamic HTML in backend(Django) and it will be parsed by Highcharts, the HTML page including the Highcharts graph will be sent to the user as an HTTP GET response.(Web app)
- JSON data will be sent as an HTTP GET response. After getting the response, mobile app will format this JSON and feed it to MPAndroidChart which will create the graph of the data.(Mobile app)

Emergency contacts will be notified by email, phone call and SMS

NON FUNCTIONAL RECQUIREMENT

PERFOMANCES RECQUIREMENTS

The steps involved to perform the implementation of garage database are as listed below

E-R DIAGRAM

The E-R Diagram constitutes a technique for representing the logical structure of a database in a pictorial manner. This analysis is the used to organize data as a relation, normalizing relation and finally obtaining a relation database.

ENTITIES: Which specify distinct real-world items in an application

PROPERTIES: Which specify properties of an entity and relationships

RELATIONSHIP: Which connect entities and represent meaningful dependencies between them:

NORMALIZATION

The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to wastage of storage space and increase in the total size of the data stored

If a database is not properly designed it can give rise to modification anomalies. Modification anomalies arise when data is added to, changed or deleted from a database table. Similarly, in traditional databases as well as improperly designed relational databases, data redundancy can be a problem. These can be eliminated by normalizing a database.

SAFETY REQUIREMENT

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

SECURITY REQUIREMENT

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully. This helps because the Patient profile and log in system are also stored in the database

SOFTWARE QUALITY ATTRIBUTES

Usability

Since user interfaces of the project does not have complex tasks on them, user should have confidence using those in no more than an hour. Usability requirements of the system are:

- Necessary configurations of fixed device should be automatic not to confuse user.
- Charts on the mobile should be simplistic and clear.
- Questions on the user feedback form should be little, short and clear. (Not more than 3-5 questions, questions should be at most two lines etc.)
- Website and mobile application interfaces should provide a uniform look and feel between all of the pages.

Reliability

- Cloud component in our system should be available at least 99.5% of the time since it should always answer requests from website and mobile phones.
- Mean time to repair the cloud server should be less than a day. System should have a failover mechanism to make the server working. (Load should be balanced between more than one servers.)

- Since we are tracking extreme situations and they can be observable in just a few seconds, wearable component should be working 99.90 % of the time. Fixed device should also be available working 99.90% of the sleep duration for the same reason.

- Wearable and fixed devices should be working all along the sleep. In the circumstances that the fixed device cannot send data to server, fixed device should cache it and send when available.

Performance

- Wearable device should be able to send all sensor data binded to it in less than a second to the fixed device. Wearable device should be keeping up with the sensor data flow speed.

- Fixed device should be able to receive data, save to database and send periodically to cloud server.

- In the extreme situations, fixed device response should be less than 3 seconds. If there is no extreme situation, data could be stored on fixed device and can be sent to cloud later.

- Number of transactions per second can be up to 100.

- The number of customers system can hold will be 10000.

- Fixed device should have a processor capable of applying exhaustive operations.

- Fixed device will not need great memory as only one users’ data will be saved and analyzed. Cloud server will need memory linearly depending on user number.

- On the fixed device, there is not a large disk space need since data will be sent to cloud with chunks and there will be one user using it. On the cloud, however there could be vast storage need. As user number gets larger, disk storage need will linearly increase.

Supportability

- Codes will be written according to GNU coding standards.

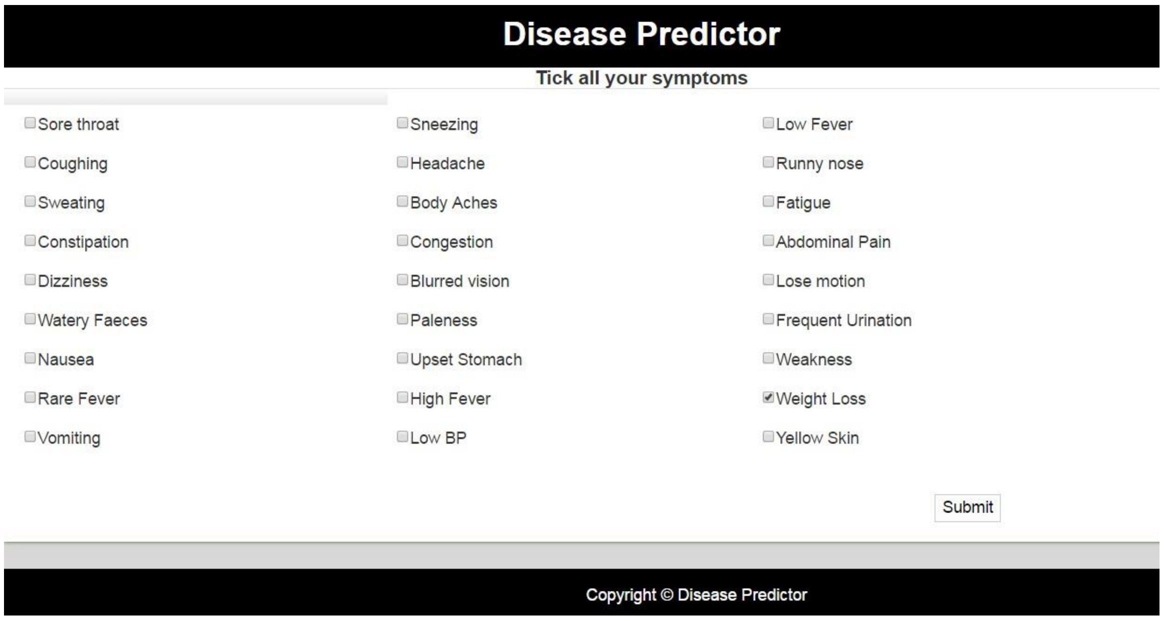
- Naming convention will be as follows:
    - Python codes in our project will have UpperCamelCase for class names,

      CAPITALIZED_WITH_UNDERSCORES for constants, and lowercase_separated_by_underscores for other names.

  - Java and C codes will have CamelCase naming convention.

- Class libraries that will be used contain but is not limited to are machine learning libraries for python, visualization libraries for python and android. Libraries can be extended for future needs.

APPENDIX

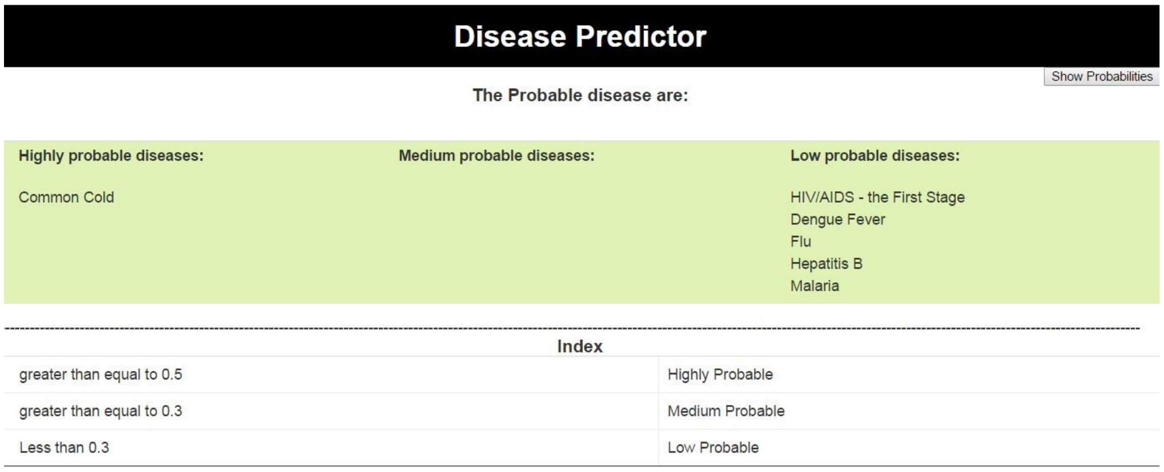


Disease Predictor

Tick all your symptoms

☐ Sore throat	☐ Sneezing	☐ Low Fever
☐ Coughing	☐ Headache	☐ Runny nose
☐ Sweating	☐ Body Aches	☐ Fatigue
☐ Constipation	☐ Congestion	☐ Abdominal Pain
☐ Dizziness	☐ Blurred vision	☐ Lose motion
☐ Watery Faeces	☐ Paleness	☐ Frequent Urination
☐ Nausea	☐ Upset Stomach	☐ Weakness
☐ Rare Fever	☐ High Fever	☑ Weight Loss
☐ Vomiting	☐ Low BP	☐ Yellow Skin

Output showing the probability of the disease



Disease Predictor

The Probable disease are:

Highly probable diseases:	Medium probable diseases:	Low probable diseases:
Common Cold		HIV/AIDS - the First Stage Dengue Fever Flu Hepatitis B Malaria

Index

greater than equal to 0.5	Highly Probable
greater than equal to 0.3	Medium Probable
Less than 0.3	Low Probable

References

IEEE Guide for Software Requirements Specifications," in IEEE Std 830-1984 , vol., no., pp.1-26, Feb. 10 1984, doi: 10.1109/IEEESTD.1984.119205,
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=278253&isnumber=6883

Appendix C of Don Widrig, Dean Leffingwell, "Managing Software Requirements: A Unified Approach," Addison-Wesley Professional, Release Date: October 1999, ISBN: 0201615932.

Marko Borazio, Eugen Berlin, Nagihan Kücükyildiz, Philipp M Scholl and Kristof Van Laerhoven, Towards Benchmarked Sleep Detection with Inertial Wrist-worn Sensing Units",

ICHI 2014, Verona, Italy, IEEE Press, 09/2014.

Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), URL: http://standards.ieee.org/getieee802/download/802.15.4d-2009.pdf