

**Project Title:** Analysis of Alert and Gait data of the residents of TigerPlace and connecting these data to find out distinctive parameter to identify the Abnormal days from Normal days.

**Submitted by:**

**Name:** Mohammad Alharbi

**Student ID:** 14084538

Department of Computer Science  
University of Missouri Columbia

**Project Supervisor:**

Dr. William Harrison  
Department of Computer Science  
University of Missouri Columbia

**Project Co-supervisor:**

Dr. Marjorie Skubic  
Dept. of Electrical & Computer Engineering  
University of Missouri Columbia

Date: July 02, 2012

# Table of Contents

1. Introduction	4
2. Related work	6
3. Background	7
4. Methods	8
4.1 Alert data processing and classification	8
4.2 Gait data processing and filter out the outliers	12
4.2.1 Gait data processing	12
4.2.2 Filter out the outliers	18
4.3 Connecting AlertDB and GaitDB	20
4.4 Analysis of the data obtained from connecting AlertDB and GaitDB	23
5. Functions of PHP Pages and Data Analysis Result	24
5.1 Functions of PHP pages	24
5.2 Data analysis result	41
6. Conclusion and Future Work	50

**Abstract:**

Over the past few decades, the number of elderly persons, who prefer living independently in unrestricted environments, have grown continuously; hence, the continuous assessment of their daily activities and detecting different signs of health status from their daily life's activities have also increased. It is anticipated that the gait data of the patients together with clinical experts' comments would help to determine the health factors which play important role in the normal or abnormal situation of the patients. To find out such health factors, i.e. gait parameters, we obtained two databases namely: AlertDB and GaitDB which contained alert and gait data of the residents of TigerPlace, an independent living facility for seniors. On the completion of the project, we successfully classified the AlertDB data into Normal and Abnormal user days and then we processed the gait data so that we can import both the DBs into one common platform, phpMyAdmin MySQL platform, hence we can access them and try different operation on them from one single platform. Then, we made relation between these two databases and after these two databases were connected, we analyzed different attributes of gait data such as the speed of walk, duration of walk etc. by taking Min, Max, Avg. of them for the Normal and Abnormal Days. We created PHP pages to show the different results in order to analyze the AlertDB and GaitDB and also to visualize graphically different parameters' behavior for Normal and Abnormal days. By analyzing the different parameters, presented graphically and statistically on the PHP pages, anyone can identify distinctive parameter i.e. health factor, if there is any, which would show certain behavior for the Abnormal user day.

**Project Title:** Analysis of Alert and Gait data of the residents of TigerPlace and connecting these data to find out distinctive parameter to identify the Abnormal days from Normal days.

## **1. Introduction**

Nowadays, there is a continuous growth of elderly persons who prefer living independently in unrestricted environments; there is also a continuous rise in the demand for continuous assessment of their daily activities and detecting different signs of health status from their daily life's activities. As the life expectancy has grown significantly over the past couple of decades and hence the lifestyle has become more active for the older persons, there is now an emphasis on determining any changes that occur in their gait patterns for the purpose of reducing the frequency of possible accidents.

I received two databases namely: AlertDB and GaitDB which contained data about the residents of TigerPlace. TigerPlace is an independent living facility for seniors designed and developed as a result of collaboration between Sinclair School of Nursing, University of Missouri and Americare Systems Inc. of Sikeston, Missouri. The primary goal of TigerPlace is to help the residents not only manage their illnesses but also stay as healthy and independent as possible. Different types of sensors (bed sensors, bathroom sensors, dining room sensors, patio sensors etc.) along with cameras were placed at different locations in the Tigerplace residence in order to acquire different types of data about the residents. These obtained data are collected and stored in database for the purpose of later analysis.

The AlertDB contains the data related to the everyday status of the residents. Clinical experts read the data from the sensors and rate the status of the residents for each day according to the alert type and alert description. The GaitDB contains the gait data of

the residents. The gait data contains different type of information about the residents such as walking speed, duration of walk, length of walk, the system time, date, height, angle of walk, quality of walk etc.

The task is to classify the AlertDB data into Normal and Abnormal user days based on the rating provided by the clinical experts. The task also includes processing the gait data that comes into text files; these data, in the text files, about the date, system time etc. are not in correct format and we have to format them correctly and also assign correct UserID for the files. After the alert data is classified and gait data are processed properly, they need to be imported into one common platform so that we can access them and try different operation on them from one single platform. I imported the data of these two databases into phpMyAdmin MySQL platform. Then, I made relation between these two databases on the basis of date. After these two databases are connected, I was able to analyze different values of gait data such as the speed of walk, length of walk etc. for the Normal and Abnormal Days. I created PHP pages to show the different results in order to analyze the AlertDB and GaitDB and also to visualize graphically different parameters' behavior for Normal and Abnormal days. By analyzing the data statistically and visually, anyone can identify a distinctive parameter, if there is any, which would show certain behavior for the Abnormal user day.

The rest of the report is organized as follows: Section 2 describes some related work which have been done in Gait assessment for older people, Section 3 discusses about some background of the work, Section 4 describes the methods which I followed to process the databases and get the output, Section 5 discusses the results which I obtained and finally, in Section 6 I have concluded.

## **2. Related work**

Many elderly persons, nowadays, prefer to live independently for as long as they are physically able to perform their daily life activities, despite the onset of conditions such as frailty and dementia. Elderly patients are particularly at-risk for late assessment of physical or cognitive changes due to many factors: their impression that such changes are simply a normal part of aging; their reluctance to admit to a problem; their fear of being institutionalized; and even the failure of physicians to fully assess their function due to the belief that no intervention is possible [1].

A number of works have been pursued in order to build a total system that will continuously monitor the activities of the senior citizens with the help of camera systems and sensors. Modern technology has given the chance to deploy such system to the elder person's residence that works silently and automatically; without interrupting their daily routine. Mihail et. al. proposed an integrated eldercare electronic health records (IEEHR) [2]. IEEHR merges health data with sensors and telehealth (vital signs) measurements. The benefit of an IEEHR system is three fold: provides physicians more tools for chronic disease management, reduces nursing workload and allows the development of health context aware algorithms for predictive health assessment [2]. According to the knowledge of the authors, none of the existing monitoring solutions has considered the health context of the resident in developing monitoring algorithms.

The work pursued by Gregory et. al. is aimed to evaluate whether residents in an independent living center such as TigerPlace, residents' family members, and healthcare providers find an interface designed to monitor resident activity levels useful and usable [3]. Rantz et. al. have developed a technological innovation that detects changes in health status that indicate impending acute illness or exacerbation

of chronic illness before usual assessment methods or self-reports of illness [4]. The authors successfully used this information in a 1-year prospective study to alert health care providers so they could readily assess the situation and initiate early treatment to improve functional independence. Another work of Gregory et. al. includes development of an early illness warning system which incorporates algorithms that issue automated alerts to clinicians, warning them of potential declines in activity levels and acute health events experienced by residents living in TigerPlace [5].

Apart from the above-mentioned works, there are much more ongoing research activities to ensure the elderly citizen's health condition. In the work, I also have worked on a particular side of this particular area of research. The work includes integration of gait data and alert data about the rating of clinical experts for each day, for each resident in TigerPlace.

### **3. Background**

Clinicians and patients interact with computers on many levels. Staff nurses and physicians collect information from their clients, store client information in clinical information systems, and then retrieve that information for making clinical decisions. The information is used by clinicians to keep an ongoing record of events, actions, behaviors, perceptions, and progress made during healthcare encounters. The effectiveness of clinical information systems that allow nursing staff to predict nursing and patient outcomes depends on the usability of the system, organizational workflow, and satisfaction with technology. Therefore, understanding human interactions with information technology has the potential to improve clinical processes and patient outcomes [3]. The clinical decisions are stored in a database which I was provided in the name of AlertDB. This database consists of several tables; each of the tables contains different information about the residents' different

time's situation. Alert from different sensors are obtained and stored for later use. This database also contains the name and type of the sensors that are plotted at different locations around the residence house, TigerPlace.

The second database, GaitDB, about which I mentioned in Section 1, contains gait pattern data of the residents. These data contains important information about the residents such as speed of walk during different times of the day, duration of walk, height of the residents, length of walk etc.

There are 4 major parts in the project. First, we need to classify the user days of AertDB into Normal or Abnormal based on the rating provided by the clinical experts. Second, we need to process the gait data so that we can use them from one platform along with the alert data; moreover we also need to filter out the visitors' data on the basis of residents' height value. Third, we need to import the alert and gait data into one common platform and connect these two databases on the basis of some common key; I used phpMyAdmin MySQL platform to access the alert and gait databases. And finally Four, we need to analyze the data in order to find any parameter (speed, duration, length etc) that would show certain pattern for the Abnormal days and hence it would be convenient for the health sector people to identify and diagnosis the health status of the elderly people. These 4 parts of the project are described in details in the following section.

## **4. Methods**

I followed the methods, described below, in order to complete the 4 major parts, described in the previous section, of the project.

### **4.1 Alert data processing and classification**

The alert data which I received was in several tables in csv format. Since, we intend to work on both the AlertDB and GaitDB from a common platform, I imported the



tables of alert data into phpMyAdmin MySQL platform. I worked mainly with 2 tables: *tblalertlog* and *feedback\_new* from the several tables of AlertDB. The *tblalertlog* contains the information such as UserID (residence id), AlertName, Occur time etc. The *feedback\_new* contains information such as alertid, rating for the resident given by clinical experts, review\_status of that alert etc. Since, in the project I worked with 10 residents, I separated the information of the 10 residents from the other residents into another table. The 10 residents have these UserID: 3004, 3013, 3017, 3037, 3038, 3046, 3047, 3048, 3049 and 3050. This table is named as, *tblAlertLogSpecificUsers*. I used the following MySQL codes for making this table:

```
CREATE TABLE `tblAlertLogSpecificUsers`  
  
SELECT * FROM `tblalertlog` WHERE UserID = 3004  
UNION  
  
SELECT * FROM `tblalertlog` WHERE UserID = 3013  
UNION  
  
SELECT * FROM `tblalertlog` WHERE UserID = 3017  
UNION  
  
SELECT * FROM `tblalertlog` WHERE UserID = 3037  
UNION  
  
SELECT * FROM `tblalertlog` WHERE UserID = 3038  
UNION  
  
SELECT * FROM `tblalertlog` WHERE UserID = 3046  
UNION  
  
SELECT * FROM `tblalertlog` WHERE UserID = 3047  
UNION  
  
SELECT * FROM `tblalertlog` WHERE UserID = 3048  
UNION  
  
SELECT * FROM `tblalertlog` WHERE UserID = 3049  
UNION  
  
SELECT * FROM `tblalertlog` WHERE UserID = 3050
```

Then, I created another table named as *tblAlertLogSpecificUsersRating* which cross match the tables: *tblAlertLogSpecificUsers* and *feedback\_new*, on the basis of *id* from *tblAlertLogSpecificUsers* and *alert\_id* from *feedback\_new*. We only considered the entries that have the 'clear' review\_status from *feedback\_new*. We ignored the entries with 'open' or 'reject' review\_status. Here is the code for creating the table *tblAlertLogSpecificUsersRating*:

```
CREATE TABLE `tblAlertLogSpecificUsersRating`

SELECT      tblAlertLogSpecificUsers.id,      tblAlertLogSpecificUsers.UserID,      tblAlertLogSpecificUsers.AlertName,
tblAlertLogSpecificUsers.Occurred, feedback_new.rating,feedback_new.review_status

FROM tblAlertLogSpecificUsers, feedback_new

WHERE tblAlertLogSpecificUsers.id = feedback_new.alert_id and feedback_new.review_status ='clear'
```

Here is the view of the table *tblAlertLogSpecificUsersRating* in Fig-1, which is generated by the above code.

id	UserID	AlertName	Occurred	rating	review_status
671	3004	Bed Restlessness	2010-11-05 00:00:00	4	clear
671	3004	Bed Restlessness	2010-11-05 00:00:00	1	clear
686	3004	Living Room	2010-11-13 00:00:00	3	clear
686	3004	Living Room	2010-11-13 00:00:00	3	clear
733	3004	Bathroom	2010-11-29 00:00:00	3	clear
733	3004	Bathroom	2010-11-29 00:00:00	4	clear
733	3004	Bathroom	2010-11-29 00:00:00	3	clear
742	3004	Living Room	2010-12-04 00:00:00	3	clear
742	3004	Living Room	2010-12-04 00:00:00	3	clear
744	3004	Slow Bed Pulse	2010-12-05 00:00:00	5	clear
744	3004	Slow Bed Pulse	2010-12-05 00:00:00	4	clear
744	3004	Slow Bed Pulse	2010-12-05 00:00:00	1	clear
748	3004	Slow Bed Breathing	2010-12-06 00:00:00	5	clear
748	3004	Slow Bed Breathing	2010-12-06 00:00:00	5	clear
748	3004	Slow Bed Breathing	2010-12-06 00:00:00	1	clear
749	3004	Slow Bed Breathing	2010-12-06 00:00:00	5	clear
749	3004	Slow Bed Breathing	2010-12-06 00:00:00	5	clear

Figure 1: Sample screenshot of *tblAlertLogSpecificUsersRating*.

Now, we need to classify the user days as Normal and Abnormal. For that purpose, I followed the following condition,

mark the entry as Abnormal, if rating > 3,

mark the entry as Normal, if rating < 3 and

mark the entry as Ignore, if rating = 3

Later, I didn't consider the entry which have 'Ignore' review\_status and filtered them out from my consideration. We executed the following MySQL codes for classifying

the data and putting them into a table named as

*tblAlertLogSpecificUsersDayStatusFinal\_1*. Here is the code:

```
CREATE TABLE `tblAlertLogSpecificUsersDayStatusFinal_1`  
SELECT UserID, Occurred, avg(rating) as Day_Avg_rating,  
CASE  
  when avg(rating)>3.0 then 'Abnormal Day'  
  when avg(rating)<3.0 then 'Normal Day'  
  when avg(rating)=3.0 then 'Ignore'  
END as Day_status  
FROM `tblAlertLogSpecificUsersRating`  
group by `UserID`,`Occurred`
```

So, finally the table, *tblAlertLogSpecificUsersDayStatusFinal\_1*, contains all the alert data classified into Abnormal, Normal and Ignore status. We will use this table later to cross match with GaitDB. The table *tblAlertLogSpecificUsersDayStatusFinal\_1* has these field: UserID, Occurred, Day\_Avg\_rating and Day\_status. The Occurred field shows the occurring time of an alert for a user. the Day\_Avg\_rating field shows the average rating for a user day. Suppose, there were multiple alerts for a user for a day; in such case, the average rating is calculated and then finally the Day\_status is calculated based on the average rating. The Day\_status field shows the status of a user day: Normal, Abnormal or Ignore.

This is the block diagram of processing the Alert data, depicted in Fig-2.

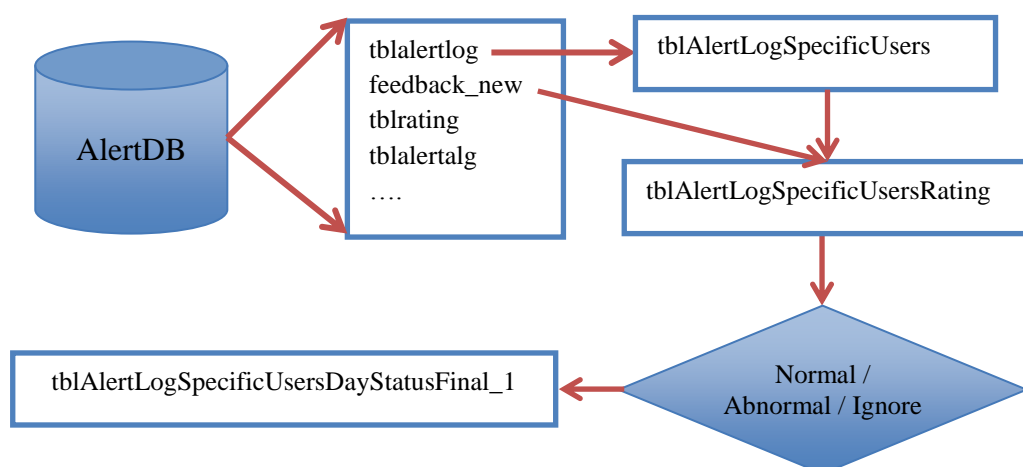


Figure 2: Block diagram of Alert data processing.

The sample screenshot of *tblAlertLogSpecificUsersDayStatusFinal\_1* is depicted in Fig-3.

UserID	Occurred	Day_Avg_rating	Day_status
3004	2010-11-05	2.5000	Normal Day
3004	2010-11-13	3.0000	Ignore
3004	2010-11-29	3.3333	Abnormal Day
3004	2010-12-04	3.0000	Ignore
3004	2010-12-05	3.3333	Abnormal Day
3004	2010-12-06	3.8571	Abnormal Day
3004	2010-12-09	4.0000	Abnormal Day
3004	2010-12-10	3.0000	Ignore
3004	2010-12-29	2.5000	Normal Day
3004	2011-01-01	1.6667	Normal Day
3004	2011-01-30	2.8333	Normal Day
3004	2011-02-06	2.5000	Normal Day
3004	2011-03-01	1.5000	Normal Day
3004	2011-03-02	3.5000	Abnormal Day
3004	2011-03-08	3.3333	Abnormal Day
3004	2011-03-21	3.3333	Abnormal Day
3004	2011-04-08	3.5000	Abnormal Day
3004	2011-04-21	2.6667	Normal Day
3004	2011-04-25	5.0000	Abnormal Day
3004	2011-04-26	4.5000	Abnormal Day
3004	2011-04-28	3.2500	Abnormal Day

Figure 3: Sample screenshot of *tblAlertLogSpecificUsersDayStatusFinal\_1*.

## 4.2 Gait data processing and filter out the outliers (visitor's data)

### 4.2.1 Gait data processing

The gait data which I received was in text files. There were more than one thousand files in total for the 10 users. Some of the data, such as the *date* and *system time*, was not in proper format. Besides, there were no *UserID* information in the text files. Since, we intend to work on both the AlertDB and GaitDB from a common platform, I processed the gait data so that they can be imported into phpMyAdmin MySQL platform. For that purpose, at first I converted the files for specific users into CSV format, then I assigned *UserID* for the files and finally, I corrected the *date* and *system time* format. The *date* field was formatted in 'yyyy-mm-dd' format so that it can be cross-matched with the *Occurred* filed of

*tblAlertLogSpecificUsersDayStatusFinal\_1* of AlertDB. Here are the steps that I followed while processing the gait data:

i) Open the 'readText\_sequence\_files\_fin.m' file in Matlab. Set the path of Matlab (from menu-> File->setpath) to the user file that you are currently working. Save the path that is set. For each user, change the path to that user's file path and remove the previous user's file path.

ii) Look at the lines of 'readText\_sequence\_files\_fin.m'.

```
userID='3004';
```

Change this variable value for each users.

```
dirName='C:\GaitData\';
```

Put the directory into this variable, dirName, where you have the GaitData.

iii) Run the matlab file; Output is saved in this format output\_3004.csv, output\_3013.csv etc.

iv) Open each of the output files and add an extra column at the beginning of the file. Put UserID for the file in that column. Save and close the file.

v) Open each files and select the second column and replace '\_' with '-'. Then format the values of this column to date (yyyy-mm-dd) format. This is the Date column.

vi) Select the 3rd column and replace '\_' with ':'. This is the system time column. Save the file and close.

vii) Now, merge all the user files into one .csv file and import into the MySQL DB.

I imported the Gait data into a table named as *tblGaitUserData*. The sample screenshot of *tblGaitUserData* is depicted in Fig-4.

UserID	Date	System_time	Height	Quality	Duration	Length	Speed	ANG	X_start	Y_start
3004	2011-10-03	16:43:26:079	63.28	0.46	3.81	86.46	22.7	2.73	24.57	95.53
3004	2011-10-03	16:47:49:829	64.02	0.5	4.14	78.11	18.87	-0.79	-55.55	159.92
3004	2011-10-03	18:28:12:370	64.64	0.51	4.41	91.7	20.8	-0.39	-60.17	128.98
3004	2011-10-03	20:31:20:561	63.31	0.85	5.2	111.35	21.39	2.73	33.45	90.9
3004	2011-10-03	20:34:01:741	64.29	0.46	3.2	79.76	24.89	-1.08	-33.92	122.04
3004	2011-10-03	21:40:56:635	62.92	0.45	4.67	76.2	16.31	0.14	-14.67	91.57
3004	2011-10-03	22:16:14:744	64.63	0.48	5.07	106.12	20.92	2.11	1.17	52.78
3004	2011-10-04	07:44:59:223	64.26	0.47	3.27	79.4	24.27	-0.91	-48.42	122.31
3004	2011-10-04	08:27:04:051	61.19	0.45	3.27	79.36	24.25	-1.06	-37.36	137.85
3004	2011-10-04	09:46:30:975	63.79	0.5	3.87	89.61	23.17	2.62	27.57	93.29
3004	2011-10-04	09:54:29:057	64.6	0.68	3.87	95.06	24.55	2.7	18.76	93.33
3004	2011-10-04	09:56:06:687	64.62	0.5	3.94	100.95	25.65	-0.36	-60.12	127.94
3004	2011-10-04	10:46:50:902	64.53	0.46	3.67	95.16	25.94	2.65	16.38	90.87
3004	2011-10-04	10:49:39:351	64.32	0.48	4.07	94.88	23.32	-0.34	-57.99	122.85
3004	2011-10-04	11:21:00:789	65.41	0.47	3	78.92	26.27	2.25	-2.78	64.03
3004	2011-10-04	12:21:07:166	59.02	0.5	2.54	72.71	28.67	-1.57	-3.66	165.71
3004	2011-10-04	16:29:04:493	64.47	0.46	3.87	93.96	24.29	-0.9	-61.65	127.55
3004	2011-10-04	18:10:23:736	63.95	0.47	3.47	79.18	22.81	-0.46	-45.42	127.94
3004	2011-10-04	20:31:53:740	64.27	0.45	3.67	87.96	23.98	2.7	14.33	93.38

Figure 4: Sample screenshot of *tblGaitUserData*.

The block diagram for processing the gait data is depicted in Fig-5.

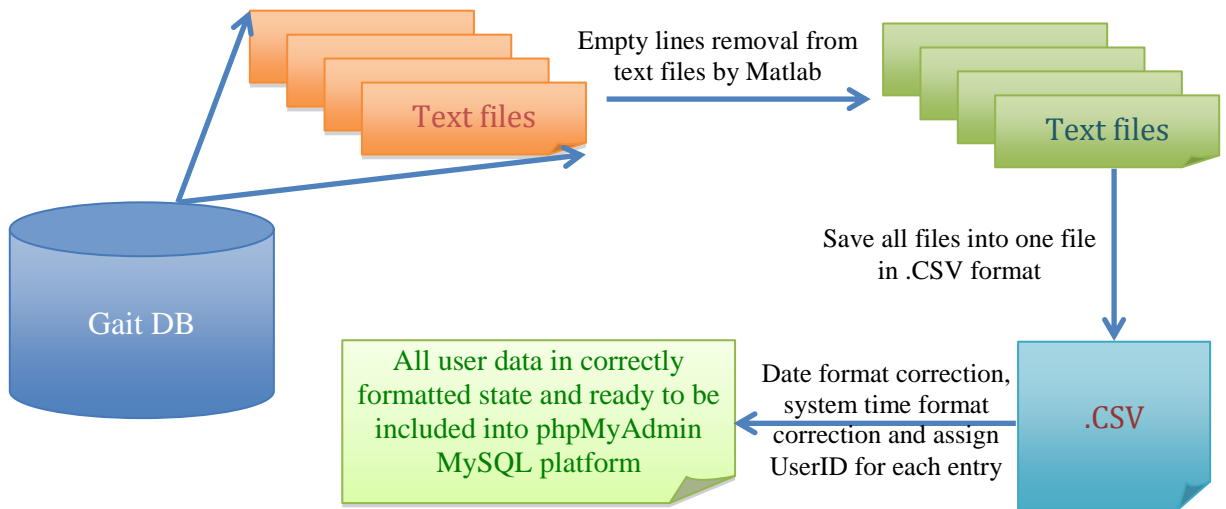


Figure 5: Block diagram of Gait data processing.

Here are the matlab codes by which we processed the Gait data in the above process.

We used two matlab files: *readtext.m* and *readText\_sequence\_files\_fin.m*. The *readText\_sequence\_files\_fin.m* calls the function *readtext( )* which is in *readtext.m*.

The function *readtext()* does all the work of processing the text files into csv file

format. Different functions of *readText\_sequence\_files\_fin.m* file does different works such as removing empty lines from the text files, reading all the text files from a folder, saving the files into .csv format etc.

-----Codes of readtext.m file-----

```
function [data, result]= readtext(text, delimiter, comment, quotes, options)

% Read (or set to default) the input arguments.
if((nargin < 1) || ~ischar(text)) % Is there a file name?
    error('First argument must be the source!');
end
opts.delimiter= ''; % Default delimiter value.
opts.comment= ''; % Default comment value.
opts.quotes= ''; % Default quotes value.
opts.file= true; % Default is file name in source.
opts.format= 'mixed'; % Default format value.
opts.op_empty= []; % Ignore empties.
opts.fname= 'text';
if(nargin >= 2), opts.delimiter= delimiter; end
if(nargin >= 3), opts.comment= comment; end
if(nargin >= 4), opts.quotes= quotes; end
if(nargin >= 5), options= options;
else
    options= '';
end

if (~ischar(opts.delimiter) || isempty(opts.delimiter))
    error('Argument "delimiter" must be a non-empty string.');
```

```
elseif(~ischar(opts.comment) || (length(opts.comment) > 1))
    error('Argument "comment" must be a string of maximum one character.');
```

```
elseif(~ischar(opts.quotes) || (length(opts.quotes) > 2))
    error('Argument "quotes" must be a string of maximum two characters.');
```

```
elseif(~ischar(options) )
    error('Argument "options" must be a string.');
```

```
end
mywaitbar= @emptywaitbar; % Default is using no waitbar ...
th= []; % ... so empty waitbar handle.
options= lower(options);
if(~isempty(strfind(options, 'textsource'))), opts.file= false; end
if(~isempty(strfind(options, 'numeric'))), opts.format= 'numeric'; end
if(~isempty(strfind(options, 'textual'))), opts.format= 'textual'; end
if(~isempty(strfind(options, 'empty2zero'))), opts.op_empty= 0; % Replace by zero
elseif(strcmp(opts.format, 'numeric') || ~isempty(strfind(options, 'empty2nan'))
    opts.op_empty= NaN; % Replace by NaN.
end
if(strcmp(opts.format, 'textual')), opts.op_empty= num2str(opts.op_empty); end % Textual 'empty'.

% Set the default return values.
result.min= Inf;
result.max= 0;
result.quote= 0;

% Read the file.
if(opts.file)
    opts.fname= text;
    [fid, errmsg]= fopen(text, 'r'); % Open the file.
    if(fid < 0), error(['Trying to open "' opts.fname '" : ' errmsg]); end
    text= transpose(fread(fid, 'uchar=>char')); % Read the file.
    fclose(fid); % Close the file.
end

if(~isempty(strfind(options, 'usewaitbar')))
    mywaitbar= @waitbar;
    th= mywaitbar(0, '(readtext) Initialising...'); % Show waitbar.
    set(findall(th, '-property', 'Interpreter'), 'Interpreter', 'none'); % No (La)TeX formatting.
end

% Clean up the text.
eol= char(10); % Using unix-style eol.
text= strrep(text, [char(13) char(10)], eol); % Replace Windows-style eol.
text= strrep(text, char(13), eol); % Replace MacClassic-style eol.
if(~isempty(opts.comment))
    text= regexprep(text, ['^\' opts.comment '[^\' eol ]*' eol], ''); % Remove entire commented lines.
    text= regexprep(text, ['\' opts.comment '[^\' eol ]*'], ''); % Remove commented line endings.
end
if(isempty(text) || text(end) ~= eol), text= [text eol]; end % End string with eol, if none.
```

```

% Find column and row dividers.
opts.delimiter= strrep(opts.delimiter, '\t', char(9)); % Convert to one char, quicker?
opts.delimiter= strrep(opts.delimiter, '\n', char(10));
opts.delimiter= strrep(opts.delimiter, '\r', char(13));
opts.delimiter= strrep(opts.delimiter, '\f', char(12));
opts.delimiter= strrep(opts.delimiter, [char(13) char(10)], eol); % Replace Windows-style eol.
opts.delimiter= strrep(opts.delimiter, char(13), eol); % Replace MacClassic-style eol.
if(1 == length(opts.delimiter)) % Find column dividers quickly.
    delimS= find((text == opts.delimiter) | (text == eol));
    delimE= delimS;
elseif isempty(regexpi(opts.delimiter, '[\+*\?\/\|\^$<>.\|]', 'once')) % Find them rather quickly.
    delimS= strfind(text, opts.delimiter);
    eols= find(text == eol);
    delimE= union(eols, delimS + length(opts.delimiter) - 1);
    delimS= union(eols, delimS);
else % Find them with regexp.
    [delimS, delimE]= regexp(text, [opts.delimiter '|' eol]);
end
divRow= [0, find(text == eol)]; % Find row dividers+last.

% Keep quoted text together.
if(~isempty(opts.quotes)) % Should we look for quotes?
    if(length(opts.quotes) == 1) || (opts.quotes(1) == opts.quotes(2)) % Opening char == ending.
        exclE= find(text == opts.quotes(1));
        exclS= exclE(1:2:end);
        exclE= exclE(2:2:end);
    else % Opening char ~= closing.
        exclS= find(text == opts.quotes(1));
        exclE= find(text == opts.quotes(2));
    end
    if(length(exclS) ~= length(exclE)) || any(exclS > exclE)
        close(th); % Close waitbar or it'll linger.
        error('Opening and closing quotes don't match in %s.', opts.fname);
    elseif(~isempty(exclS)) % We do have quoted text.
        mywaitbar(0, th, 'readtext Doing quotes...'); % Inform user.
        r= 1;
        rEnd= length(exclS);
        n= 1;
        nEnd= length(delimS);
        result.quote= rEnd;
        exclS(end+1)= 0; % This and next lines are needed in cases with only one quote in text.
        exclE(end+1)= 0;
        while((n <= nEnd) && (r <= rEnd)) % "Remove" delimiters and newlines within quotes.
            while((r <= rEnd) && (delimS(n) > exclE(r)), r= r+1; end % Next end-quote after newline.
            while((n <= nEnd) && (delimS(n) < exclS(r)), n= n+1; end % Next newline after start-quote.
            while((n <= nEnd) && (delimS(n) >= exclS(r)) && (delimS(n) <= exclE(r))
                delimS(n)= 0; % Newlines inside quote.
                n= n+1;
            end
            mywaitbar(n/nEnd); % Update waitbar.
        end
        mywaitbar(1);
        delimE= delimE(delimS > 0);
        delimS= delimS(delimS > 0);
    end
end
delimS= delimS-1; % Last char before delimiter.
delimE= [1 delimE(1:end-1)+1]; % First char after delimiter.

% Do the stuff: convert text to cell (and maybe numeric) array.
mywaitbar(0, th, sprintf('readtext Processing "%s"...', opts.fname));
r= 1;
c= 1; % Pre-size data to optimise speed.
data= cell(length(divRow), ceil(length(delimS)/(length(divRow)-1)));
nums= zeros(size(data)); % Pre-size nums to optimise speed.
nEnd= length(delimS); % Prepare for a waitbar.
istextual= strcmp(opts.format, 'textual');
for n=1:nEnd
    temp= strtrim(text(delimE(n):delimS(n))); % Textual item.
    data{r, c}= temp; % Textual item.
    if(~istextual)
        lenT= length(temp);
        if(lenT > 0)
            % Try to get 123, 123i, 123i + 45, or 123i - 45
            [a, count, errmsg, nextindex]= sscanf(temp, '%f%1[ij] %1[+-] %f', 4);
            if(isempty(errmsg) && (nextindex > lenT))
                if (count == 1), nums(r, c)= a;
                elseif(count == 2), nums(r, c)= a(1)*i;
                elseif(count == 4), nums(r, c)= a(1)*i + (44 - a(3))*a(4);
                else
                    nums(r, c)= NaN;
                end
            elseif(regexpi(temp, '[^0-9eij \. \+ \-]', 'once')) % Remove non-numbers.
                nums(r, c)= NaN;
            else

```



```

        nums(r, c)=    s2n(temp, lenT);          % Some other kind of complex number.
    end
    else
        nums(r, c)= NaN;
    end
end
if(text(delimS(n)+1) == eol)                    % Next row.
    result.min=    min(result.min, c);          % Find shortest row.
    result.max=    max(result.max, c);          % Find longest row.
    r=            r+1;
    c=            0;
    if(bitand(r, 15) == 0), mywaitbar(n/nEnd); end % Update waitbar.
end
c=            c+1;
end

.....

% Try to get 123 + i*45 or 123 - i*45
[a,count,errmsg,nextindex] = sscanf(s, '%f %1[+-] %1[ij] %1[*] %f',5);
if isempty(errmsg) && (nextindex > lenS))
    if(count == 5),        x=    a(1) + (44 - a(2))*i*a(5);
    end
    return
end

% None of the above cases.
x=    NaN;

end

function emptywaitbar(varargin)
end
-----End of codes of readtext.m file---

```

#### ----Codes for readText\_sequence\_files\_fin.m-----

```

userID='3046';
dirName='/Users/abmtariqislam/Downloads/GaitData/';
finDir = strcat(dirName,userID);

dirInfo = dir(fullfile(finDir, '*.txt'));      %# Get structure of directory information
isDir = [dirInfo.isdir];                    %# A logical index the length of the
                                            %# structure array that is true for
                                            %# structure elements that are
                                            %# directories and false otherwise
dirNames = {dirInfo(isDir).name};           %# A cell array of directory names
fileNames = {dirInfo(~isDir).name};         %# A cell array of file names

Length=length(fileNames)

for k= 1:1:Length

    fileName=strcat(Str1,num2str(k),Str2);

    [a,b]=readtext(char(fileNames(k)), '[, \t]', '#', ' ', 'textual-empty2zero');

    %[a,b]=readtext(walkData-01_02_2012.txt, '[, \t]', '#', ' ', 'textual-empty2zero');

%remove zero containing rows
[r,c]=size(a);
test = true(r,1);
for i=r:-1:1
    if isequal(a{i,2}, '0')
        test(i,1)=false;
    end
end
a = a(test,:);

%% write data into CSV file
[m,n]=size(a);

CC = cell(m,n+1);
CC(:,1:2:end) = a;
CC(:,2:2:end,:) = {' '};

```

```

CC = arrayfun(@(i) [CC{i,:}], 1:m, 'UniformOutput',false);    '%'

%# write lines to file
%fid = fopen('output.csv','wt');
fid = fopen('output_1.csv','a+');
fprintf(fid, '%s\n',CC{:});
fclose(fid);

outFile1= 'output_';
outFile2= '.csv';
outFileFin=strcat(outFile1,userID,outFile2);

fid = fopen(outFileFin,'a+');
fprintf(fid, '%s\n',CC{:});
fclose(fid);
end
% end
----End of codes for readText_sequence_files_fin.m-----

```

#### 4.2.1 Filter out the outliers (visitors' data)

The gait data, which was provided, contained both the resident's and visitor's data. In order to analyze the resident's data, we need to filter out the visitor's data i.e. the outliers. I took the *height* value as a measuring parameter by which we can detect the outliers and remove them from the data. Since the height of visitors will be different from the height of the resident, it is a good approach to filter out the visitors on the basis of *height* value. I took the entries for previous 7 days (we can take any number of days: 7 or 14 or 21 etc.) in order to filter out the outliers for the selected date. For example, we need to find out who are the outliers for the date 2011-11-01. I followed the formula in Equation.1 below to find out the outliers:

$$(\text{Height} - \text{Mean}) / \text{Standard Deviation} \dots \dots \dots (1)$$

Here, Mean and Standard Deviation are for the previous 7 days + the current date which we selected. For example, we selected date: 2011-11-01 for user 3004. So, the Mean is the Mean of Height values of previous 7 days + the current day (2011-11-01) and Standard Deviation is also similar.

Now, to determine which is the outlier, let say, there a Height value 64.64. So, for the Equation.1:

Height= 64.64, Mean =63.13, Standard Deviation = 2.38 (Mean and Standard Dev. Is calculated before.)

According to the Equation.1, we get this value:  $(64.64 - 63.13)/2.38 = 0.63$

Now, we have to choose a distance value; this distance value will determine which values will be inside our consideration zone and which ones will be the outliers.

Suppose, we pick the distance value as 1. Then if, the calculated value from Equation.1 is greater than this distance value (here, 1) then all the data in that row with that height value will be considered as outlier. So, let's say, we choose the Distance = 1. So, outliers are the values which follows the formula in Equation.2:

$$\text{Outliers} = \text{Absolute} ( (\text{Height}- \text{Mean}) / \text{Standard Deviation}) > \text{Distance} \dots \dots (2)$$

We got the **(Height- Mean) / Standard Deviation** = 0.63 and Distance = 1. Now, we compared the value  $0.63 > 1$  ; No this is smaller than 1. So, this is not an outlier. So, we will consider the data on the row which have the height value = 64.64.

In the Equation.2, we took the absolute value, because the **(Height- Mean) / Standard Deviation** value could be negative; hence we took the absolute to make it positive and then compare to the Distance value.

Now, let's take another Height value: 59.07. According to Equation.1, we get this value:  $(59.07-63.13)/2.38 = -1.71$ . According to the Equation.2:  $\text{Absolute} (-1.71) > 1 = 1.71 > 1$ ; Yes 1.71 is greater than 1. So, the Height=59.07 is a outlier, hence all he data on this height row will also be outlier data. So, we will *not* consider these height's data and mark it as outlier.

I performed the following piece of code in MySQL for detecting the outliers:

```
SELECT Date, System_time, Height, Speed, agro.mean as Avg, agro.dev as Dev, (Height - agro.mean) /
agro.dev AS num_devs FROM tblGaitUserData_Day_status CROSS JOIN ( SELECT AVG(Height) AS
mean, STDDEV(Height) AS dev FROM tblGaitUserData_Day_status Where UserID = 3004 and Date
between Date_Add(str_to_date('2012-02-15','%Y-%m-%d'),INTERVAL -7 Day) and str_to_date('2012-02-
15','%Y-%m-%d')) agro WHERE UserID = 3004 and Date = str_to_date('2012-02-15','%Y-%m-%d') and
ABS( Height - agro.mean ) / agro.dev > 1 ORDER BY Date, System_time
```

In order to show the data without the outliers, I used the following MySQL codes:

```
SELECT t.System_time, t.Height, t.Length, t.Duration, t.Speed, t.Day_status from
tblGaitUserData_Day_status t left join (SELECT tblGaitUserData_Day_status.System_time,
tblGaitUserData_Day_status.Height FROM tblGaitUserData_Day_status CROSS JOIN ( SELECT
AVG(Height) AS mean, STDDEV(Height) AS dev FROM tblGaitUserData_Day_status Where UserID = 3004
and Date between Date_Add(str_to_date('2011-11-22','%Y-%m-%d'),INTERVAL -7 Day) and
str_to_date('2011-11-22','%Y-%m-%d')) agro WHERE UserID = 3004 and Date = str_to_date('2011-11-
22','%Y-%m-%d') and ABS( tblGaitUserData_Day_status.Height - agro.mean ) / agro.dev > 1 ORDER BY
tblGaitUserData_Day_status.Date, tblGaitUserData_Day_status.System_time) o on t.Height=o.Height
where t.UserID = 3004 and t.Date = str_to_date('2011-11-22','%Y-%m-%d') and o.Height is null
```

### 4.3 Connecting AlertDB and GaitDB

After importing both the AlertDB and GaitDB into phpMyAdmin MySQL platform, we need to make a connection between these two DBs on the basis of a common key. I used the *Occurred* field from AlertDB that is actually the date of occurring an alert and *Date* from GaitDB to connect these two DBs. From the AlertDB, we already know which dates are *Normal* and which are *Abnormal*; we need to assign the value Normal or Abnormal to the Day\_status field of the resulting table which is generated by connecting the two DBs. The status of the entries of GaitDB is not known, so we need to assign this *Normal/Abnormal Day* value to the entries whose date matches with the entries in AlertDB. The entries whose date doesn't match with AlertDB's entry, we simply assign them *Normal Day* status. Let's use an example: let's say we have these dates in AlertDB for user 3004 which are identified as Normal or Abnormal Day:

Table 1: Sample data for AlertDB

UserID	Occurred	Day_Avg_rating	Day_status
3004	2011-11-01	2.5	Normal Day
3004	2011-12-10	1	Normal Day
3004	2012-01-15	4.5	Abnormal Day
3004	2012-01-18	4	Abnormal day

And, GaitDB has the following data for the following dates:

Table 2: Sample data for GaitDB

UserID	Date	Sysem_time	Height	Quality	Duration	Length	Speed	ANG	X_start	Y_start
3004	2011-11-01	09:30:12:019	65.54	0.5	3.21	89.87	22.7	2.73	24.56	45.67
3004	2011-11-15	10:30:12:019	66.34	0.6	6.54	88.87	25.8	-0.93	33.45	139.50
3004	2011-12-10	19:40:12:019	67.78	0.8	2.81	78.89	23.87	0.76	56.78	158.87
3004	2011-12-16	08:30:12:017	67.45	0.5	6.75	88.88	27.87	-1.78	-34.45	30.67
3004	2012-01-14	21:20:12:025	66.68	0.9	5.67	89.90	28.89	2.87	56.67	46.57
3004	2012-01-15	17:05:16:117	68.78	0.54	6.54	85.56	24.0	3.14	45.56	129.78
3004	2012-01-16	23:09:28:034	67.45	0.67	5.78	87.78	30.34	2.24	44.30	-67.12
3004	2012-01-18	21:20:12:025	66.68	0.9	5.67	89.90	28.89	2.87	56.67	46.57
3004	2012-01-19	09:30:12:019	65.54	0.5	3.21	89.87	22.7	2.73	24.56	45.67
3004	2012-01-20	10:30:12:019	66.34	0.6	6.54	88.87	25.8	-0.93	33.45	139.50

So, when we connected these two tables, Table 1 and Table 2, at first, the rows of GaitDB were assigned *Normal/Abnormal Day* if there were matched for the dates. If there were no match for the date, that row was assigned *Normal Day* status. I used the following MySQL codes to connect the table *tblAlertLogSpecificUsersDayStatusFinal\_1* from AlertDB and *tblGaitUserData* and generate a new table *tblGaitUserData\_Day\_status*:

```
SELECT      tblGaitUserData.UserID,      tblGaitUserData.Date,      tblGaitUserData.System_time,
tblGaitUserData.Height, tblGaitUserData.Quality, tblGaitUserData.Duration, tblGaitUserData.Length,
tblGaitUserData.Speed, tblGaitUserData.ANG, tblGaitUserData.X_start, tblGaitUserData.Y_start,
ifnull(tblAlertLogSpecificUsersDayStatusFinal_1.Day_status, 'Normal Day') as `Day_status`
FROM tblGaitUserData left join tblAlertLogSpecificUsersDayStatusFinal_1 on
      tblGaitUserData.UserID = tblAlertLogSpecificUsersDayStatusFinal_1.UserID and
      tblGaitUserData.Date = tblAlertLogSpecificUsersDayStatusFinal_1.Occurred and
```

```
lower(tblAlertLogSpecificUsersDayStatusFinal_1.Day_status) != 'Ignore'
ORDER BY tblGaitUserData.UserID, tblGaitUserData.Date, tblGaitUserData.System_time
```

After connecting two tables, Table 1 and Table 2, we get the following table:

Table 3: Sample data after connecting AlertDB and GaitDB

UserID	Date	Sysem_time	Height	Quality	Duration	Length	Speed	ANG	X_start	Y_start	Days_status
3004	2011-11-01	09:30:12:019	65.54	0.5	3.21	89.87	22.7	2.73	24.56	45.67	Normal Day
3004	2011-11-15	10:30:12:019	66.34	0.6	6.54	88.87	25.8	-0.93	33.45	139.50	Normal Day
3004	2011-12-10	19:40:12:019	67.78	0.8	2.81	78.89	23.87	0.76	56.78	158.87	Normal Day
3004	2011-12-16	08:30:12:017	67.45	0.5	6.75	88.88	27.87	-1.78	-34.45	30.67	Normal Day
3004	2012-01-14	21:20:12:025	66.68	0.9	5.67	89.90	28.89	2.87	56.67	46.57	Normal Day
3004	2012-01-15	17:05:16:117	68.78	0.54	6.54	85.56	24.0	3.14	45.56	129.78	Abnormal Day
3004	2012-01-16	23:09:28:034	67.45	0.67	5.78	87.78	30.34	2.24	44.30	-67.12	Normal Day
3004	2012-01-18	21:20:12:025	66.68	0.9	5.67	89.90	28.89	2.87	56.67	46.57	Abnormal Day
3004	2012-01-19	09:30:12:019	65.54	0.5	3.21	89.87	22.7	2.73	24.56	45.67	Normal Day
3004	2012-01-20	10:30:12:019	66.34	0.6	6.54	88.87	25.8	-0.93	33.45	139.50	Normal Day

The diagram in Fig-6 shows the process of connecting the AlertDB and GaitDB. It shows the table *tblAlertLogSpecificUsersDayStatusFinal\_1* as the final output for AlertDB and *tblGaitUserData* as output for GaitDB; then it shows the connected table *tblGaitUserData\_Day\_status* as the final output table.

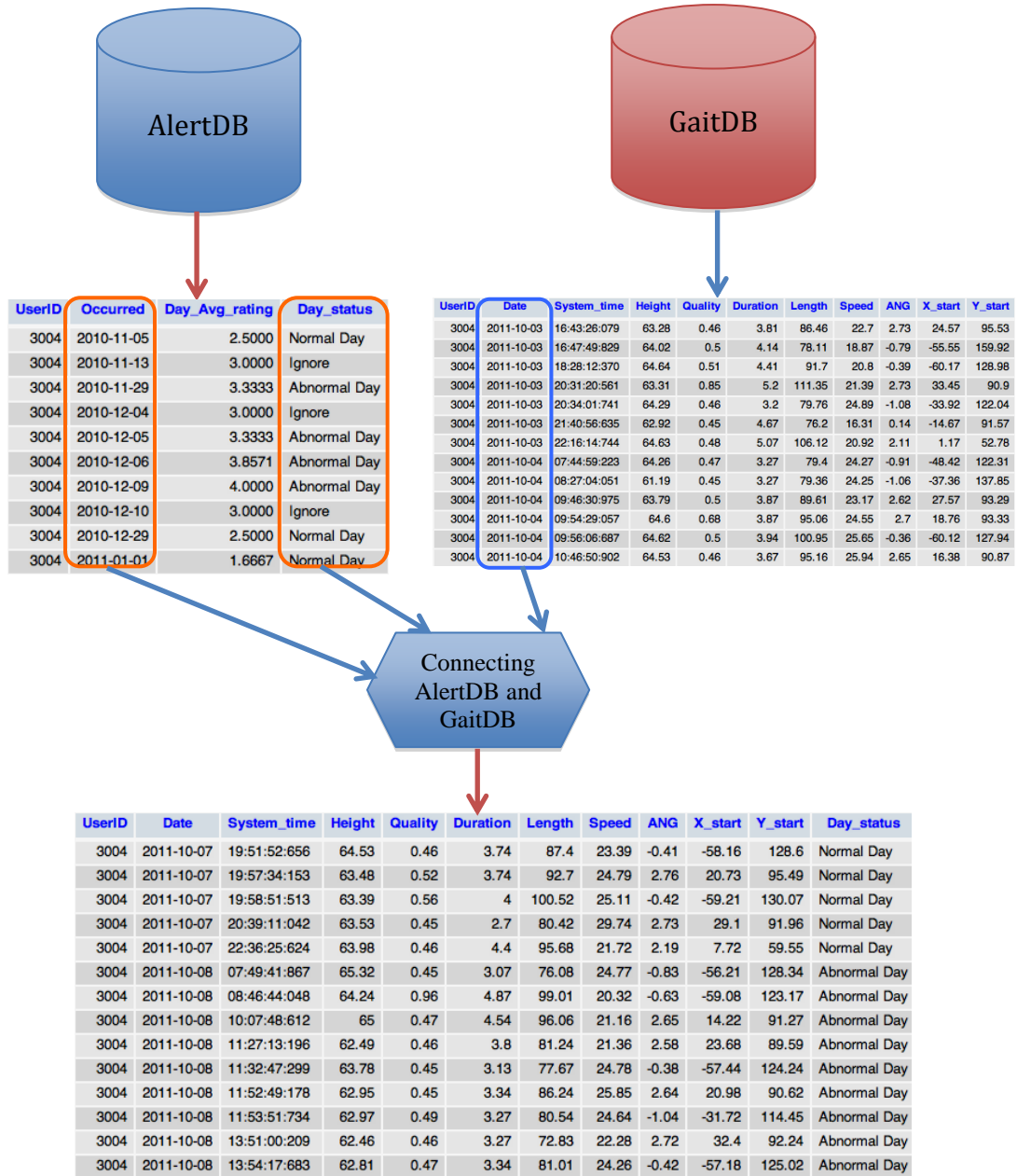


Figure 6: Block diagram of connecting AlertDB and GaitDB.

#### 4.4 Analysis of the data obtained from connecting AlertDB and GaitDB

After the two databases, Alert and Gait, are related based on the common key, the task is to analyze the resulting data which is obtained from this connection. We need to calculate the Max, Min, Avg (Mean) etc. measures for each user day in order to analyze the data. The goal of this analysis is to find out any parameter, such as speed or duration or length, which shows certain behavior for the *Abnormal* days, but shows

usual behavior for the *Normal* days. We can find out such parameter, if there exists any, by means of analyzing the different measures for the data. I executed the following MySQL codes to get the Max, Min, Avg etc. values for the user days; I have filtered out the outliers while calculating the Max, Min, Avg etc. values:

```
SELECT min(t.Speed), max(t.Speed), avg(t.Speed), stddev(t.Speed), max(t.Speed)-avg(t.Speed) from
tblGaitUserData_Day_status t left join (SELECT tblGaitUserData_Day_status.System_time,
tblGaitUserData_Day_status.Speed, tblGaitUserData_Day_status.Height FROM
tblGaitUserData_Day_status CROSS JOIN ( SELECT AVG(Height) AS mean, STDDEV(Height) AS dev
FROM tblGaitUserData_Day_status Where UserID = 3004 and Date between Date_Add(str_to_date('2011-
11-22','%Y-%m-%d'),INTERVAL -7 Day) and str_to_date('2011-11-22','%Y-%m-%d')) agro WHERE UserID
= 3004 and Date = str_to_date('2011-11-22','%Y-%m-%d') and ABS( tblGaitUserData_Day_status.Height -
agro.mean ) / agro.dev > 1 ORDER BY tblGaitUserData_Day_status.Date,
tblGaitUserData_Day_status.System_time) o on t.Height=o.Height where t.UserID = 3004 and t.Date =
str_to_date('2011-11-22','%Y-%m-%d') and o.Height is null;
```

I have designed several PHP pages to show different output related to the project. I have described the functions of these pages in brief in the following section.

## 5. Functions of PHP Pages and Data Analysis Result

In this section, I will discuss mainly about two points: first, I will briefly discuss about the functions of the PHP pages which I designed and coded to show different output related to the project and second, I will discuss about the analysis result which is obtained.

### 5.1 Functions of PHP pages

#### 5.1.1 The Home page

The home page shows the links to the 9 PHP pages which perform different tasks related to the project. The link for the home page is: <http://localhost/home.php>. There is brief description for each option (pages) just below each of the options. Here is a screenshot of the home page in Fig-7:



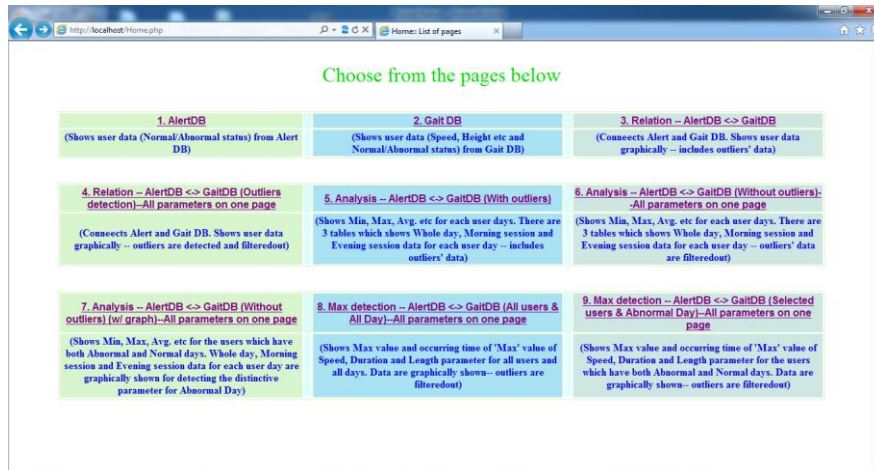


Figure 7: Sample screenshot of the *Home* page.

### 5.1.2 The *AlertDB* page

This page shows the alert data of the users. It also shows the classification results of the user days. The link for this page is: [http://localhost/search\\_alertdb.php](http://localhost/search_alertdb.php). The outlook of this page is shown in Fig-8:

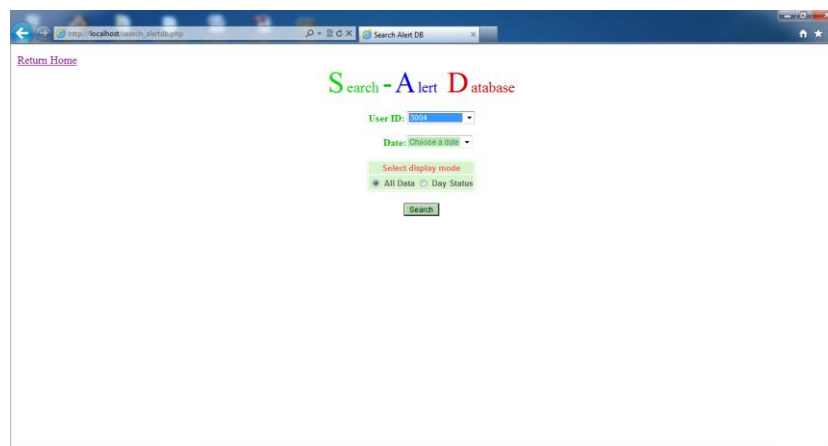
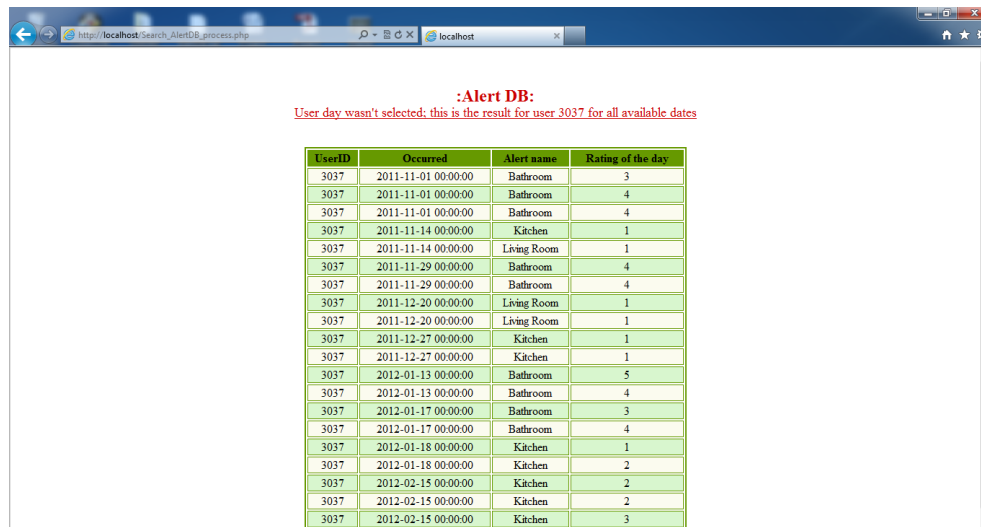


Figure 8: Outlook of the *AlertDB* page.

Anyone can select a user from the *UserID* dropdown list and then, if he/she press the *All Data* option, he/she can see every data, including multiple entries for a date. If he/she chooses the *Day Status* option, then the status of the days for the selected user will be displayed. If a specific date is selected from the *Date* dropdown list, then he/she can see either *All Data* or *Day Status* for that selected date.

To have a better idea about the All Data and Day Status option, let's see the following outputs in Fig-9 and Fig-10. For user 3007, when I selected the *All Data* option, I got this output in Fig-9:

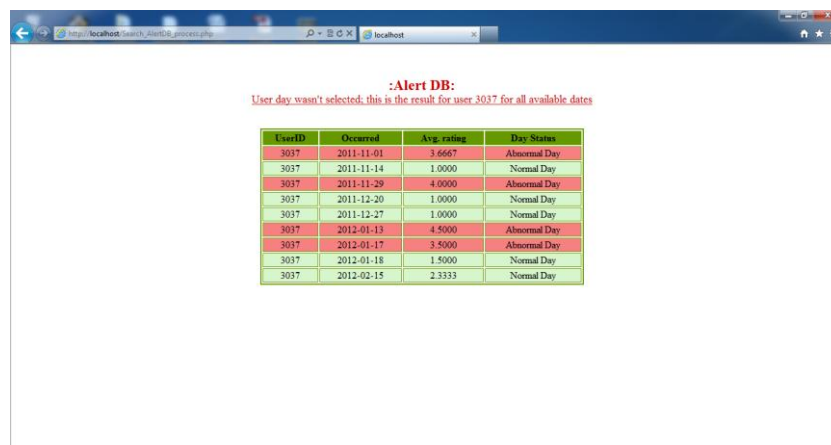


**:Alert DB:**  
User day wasn't selected; this is the result for user 3037 for all available dates

UserID	Occurred	Alert name	Rating of the day
3037	2011-11-01 00:00:00	Bathroom	3
3037	2011-11-01 00:00:00	Bathroom	4
3037	2011-11-01 00:00:00	Bathroom	4
3037	2011-11-14 00:00:00	Kitchen	1
3037	2011-11-14 00:00:00	Living Room	1
3037	2011-11-29 00:00:00	Bathroom	4
3037	2011-11-29 00:00:00	Bathroom	4
3037	2011-12-20 00:00:00	Living Room	1
3037	2011-12-20 00:00:00	Living Room	1
3037	2011-12-27 00:00:00	Kitchen	1
3037	2011-12-27 00:00:00	Kitchen	1
3037	2012-01-13 00:00:00	Bathroom	5
3037	2012-01-13 00:00:00	Bathroom	4
3037	2012-01-17 00:00:00	Bathroom	3
3037	2012-01-17 00:00:00	Bathroom	4
3037	2012-01-18 00:00:00	Kitchen	1
3037	2012-01-18 00:00:00	Kitchen	2
3037	2012-02-15 00:00:00	Kitchen	2
3037	2012-02-15 00:00:00	Kitchen	2
3037	2012-02-15 00:00:00	Kitchen	3

Figure 9: Outlook of the *AlertDB* page with *All Data* option chosen.

From Fig-9, we can see that, for date 2011-11-01, we have 3 entries and we can see all 3 of them. To calculate the day status, we made average of the rating value of these 3 entries. By choosing *Day status* option, we can see the status of the dates for a user. Fig-10 shows such output when this option is selected:



**:Alert DB:**  
User day wasn't selected; this is the result for user 3037 for all available dates

UserID	Occurred	Avg. rating	Day Status
3037	2011-11-01	3.6667	Abnormal Day
3037	2011-11-14	1.0000	Normal Day
3037	2011-11-29	4.0000	Abnormal Day
3037	2011-12-20	1.0000	Normal Day
3037	2011-12-27	1.0000	Normal Day
3037	2012-01-13	4.5000	Abnormal Day
3037	2012-01-17	3.5000	Abnormal Day
3037	2012-01-18	1.5000	Normal Day
3037	2012-02-15	2.3333	Normal Day

Figure 10: Outlook of the *AlertDB* page with *Day Status* option chosen.

From Fig-10, we can see all the dates for user 3037 with day status. If we look at the first entry for date: 2011-11-01, we can see the average rating, which is 3.67; we got by taking average of the 3 entries (look at the Fig-9).

### 5.1.3 The *GaitDB* page

This page shows the gait data of the users. I received the gait data in text format and then formatted and converted into csv format and then imported into MySQL platform. The link for this page is: [http://localhost/search\\_gaitdb.php](http://localhost/search_gaitdb.php). The outlook of this page is shown in Fig-11:



Figure 11: Outlook of the *GaitDB* page.

This page also shows two options for data of the users. If anyone selected *All Data*, then he/she can view all the gait data for all the available dates in the gait database with the day status. If anyone selected *Day Status*, then he/she can view the day status for all the available dates in the gait database. For example, if we choose to see the status of all available dates for the user 3050, then we need to select the user 3050 from the UserID dropdown list and then, we need to select the *Day Status* option and then we can see the output which is depicted in Fig-12.

**:Gait DB:**  
*User day wasn't selected; this is the result for user 3050 for all available dates*

UserID	Date	Day status
3050	2012-01-01	Normal Day
3050	2012-01-02	Normal Day
3050	2012-01-03	Normal Day
3050	2012-01-04	Normal Day
3050	2012-01-05	Normal Day
3050	2012-01-06	Normal Day
3050	2012-01-07	Normal Day
3050	2012-01-08	Normal Day
3050	2012-01-09	Normal Day
3050	2012-01-10	Normal Day
3050	2012-01-11	Normal Day
3050	2012-01-12	Normal Day
3050	2012-01-13	Normal Day
3050	2012-01-14	Normal Day
3050	2012-01-15	Normal Day
3050	2012-01-16	Normal Day
3050	2012-01-17	Normal Day
3050	2012-01-18	Normal Day
3050	2012-01-19	Normal Day
3050	2012-01-20	Normal Day

Figure 12: Outlook of the *GaitDB* page with *Day Status* option chosen.

#### 5.1.4 The Relation -- *AlertDB* <-> *GaitDB* page

This is the page that shows the relation between the two databases after they are connected based on common key. This page shows the data (Speed, Duration, Length, Height) graphically. The output graph is drawn like this: along the y-axis there is *Speed/Duration/Length/Height* data and along the x-axis there is *System time* data. This page includes all the data (includes outliers or visitors). This page is created just to make sure, if someone wants to see all the data (including the outliers) graphically, then he/she can see them from this page. The link for this page is: [http://localhost/s\\_alert\\_gait\\_wgraph.php](http://localhost/s_alert_gait_wgraph.php). The image in Fig-13 shows the sample snapshot of this page:

**Relation - (Alert DB) <-> (Gait DB)**

User ID:

Date:

Select graph parameter for Y axis

☒ Speed (length/duration)
 ☐ Duration (time elapsed during the walk in sec)
 ☐ Length (length of the walk in inches)
 ☐ Height (estimated height of the person that walked)

Figure 13: Outlook of the *Relation -- AlertDB <-> GaitDB* page.

Now, after selecting any date from the date dropdown box, if we choose any of the 4 options (for example, we clicked the Speed), and after clicking the search button, we will see this page as in Fig-14:

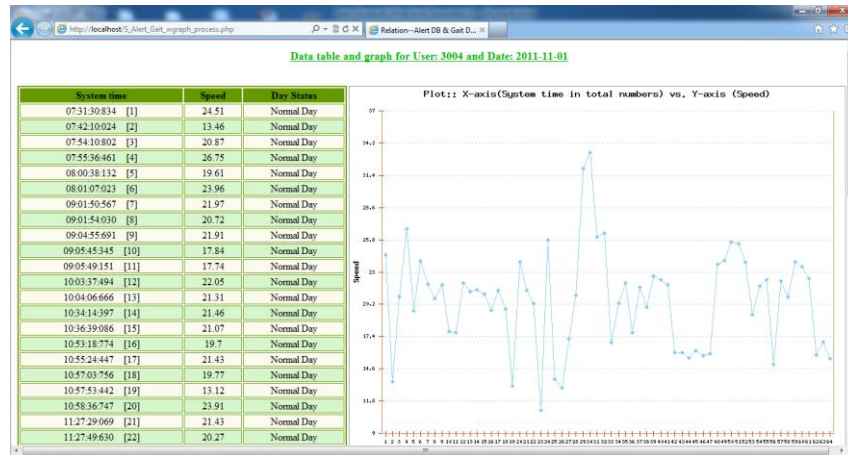


Figure 14: Outlook of the *Relation -- AlertDB <-> GaitDB* page with Speed option chosen.

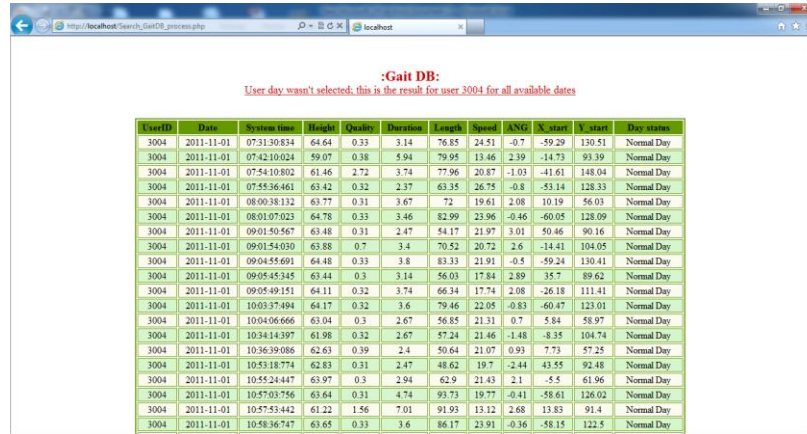
This page, in Fig-14, shows the data for the selected user for the speed variable and it also shows the system time and day status for that date. It shows the graph which is a plot of *System\_time* vs *Speed*. *System\_time* is shown with index number because it's less messy to plot the index number of the *System\_time* rather than the actual *System\_time*. Now, if we don't select the 4 options and click on search after selecting the date, the sample result will be similar to the image in Fig-15:

Query result for User: 3004 and Date: 2011-11-22

Rowid	Date	System time	Height	Quality	Distance	Length	Speed	ASGI	X-axis	Y-axis	Day Status
3004	2011-11-22	07:41:05176	64.2	0.32	4.2	96.92	23.06	-0.52	-62.71	135.33	Normal Day
3004	2011-11-22	07:41:16377	63.13	0.3	3.34	65.8	19.72	-2.59	65.39	101.86	Normal Day
3004	2011-11-22	08:20:21869	64.51	0.47	6.28	129.1	20.57	2.05	3.02	55.29	Normal Day
3004	2011-11-22	08:24:18897	63.93	0.34	5.94	95.58	16.08	-0.74	-55.94	156.26	Normal Day
3004	2011-11-22	09:01:44348	63.71	0.31	2.6	50.28	19.14	2.85	38.03	89.91	Normal Day
3004	2011-11-22	09:01:46875	65.21	2.51	2.47	50.78	20.58	2.58	-21.23	103.99	Normal Day
3004	2011-11-22	09:05:32059	63.12	0.33	4.64	97.14	20.93	-0.54	-60.68	137.75	Normal Day
3004	2011-11-22	09:18:14692	63.54	0.32	2.4	55.89	22.96	-3.05	55.18	96.88	Normal Day
3004	2011-11-22	09:18:17827	63.48	1.96	3.07	77.71	25.33	2.48	-14.12	96.6	Normal Day
3004	2011-11-22	09:19:42316	64.05	0.34	3.8	91.89	24.1	-0.37	-56.34	124.06	Normal Day
3004	2011-11-22	09:21:56710	64.48	0.34	2.27	50.53	22.27	-2.59	54.55	96	Normal Day
3004	2011-11-22	11:02:43474	62.45	0.39	3.27	52.21	15.98	1.11	3.92	46.05	Normal Day
3004	2011-11-22	11:12:07261	63.35	0.34	2.87	61.34	21.39	-2.49	61.56	106.32	Normal Day
3004	2011-11-22	11:12:17338	62.97	0.3	3.6	74.91	20.79	2.08	13.42	52.85	Normal Day
3004	2011-11-22	11:13:53644	63.92	0.31	2.94	54.97	18.72	-0.72	-59.17	144.57	Normal Day
3004	2011-11-22	11:15:56569	63.12	0.3	3	58.82	19.61	0.07	-4.49	94.81	Normal Day
3004	2011-11-22	11:23:01899	62.44	0.31	2.47	51.09	20.7	-3.08	52.78	98.87	Normal Day
3004	2011-11-22	11:22:05782	62.44	1.89	3.8	71.08	18.69	2.2	-6.65	105.14	Normal Day
3004	2011-11-22	16:22:15574	63.94	0.34	2.8	52.24	18.63	-0.6	-61.48	127.82	Normal Day
3004	2011-11-22	16:22:18710	63.37	0.4	3.07	55.78	18.19	-1.11	-16.21	88.87	Normal Day
3004	2011-11-22	16:50:38652	60.35	0.45	3.67	58.91	16.05	0.52	20.99	80.03	Normal Day

Figure 15: Outlook of the *Relation -- AlertDB <-> GaitDB* page with no option chosen.

Now, if we don't select any 4 options and don't also select any date and we have clicked the search then all the data about that user will be shown, as the image in Fig-16:



**:Gait DB:**  
User day wasn't selected, this is the result for user\_3004 for all available dates

UserID	Date	System time	Height	Quality	Duration	Length	Speed	ANG	X_start	Y_start	Day status
3004	2011-11-01	07:51:30.834	64.64	0.33	3.14	76.85	24.51	-0.7	-59.29	130.51	Normal Day
3004	2011-11-01	07:52:10.021	59.07	0.38	5.94	79.95	13.46	2.39	-14.73	93.39	Normal Day
3004	2011-11-01	07:54:10.802	61.46	2.72	3.74	77.96	20.87	-1.03	-41.61	148.04	Normal Day
3004	2011-11-01	07:55:36.461	63.42	0.32	2.37	63.35	26.75	-0.8	-53.14	128.33	Normal Day
3004	2011-11-01	08:00:38.132	63.77	0.11	3.67	72	19.61	2.08	10.19	56.03	Normal Day
3004	2011-11-01	08:01:07.023	64.78	0.33	3.46	82.99	23.96	-0.46	-60.05	128.09	Normal Day
3004	2011-11-01	09:01:50.567	63.48	0.31	2.47	54.17	21.97	3.01	50.46	90.16	Normal Day
3004	2011-11-01	09:01:54.030	63.88	0.7	3.4	70.52	20.72	2.6	-14.41	104.05	Normal Day
3004	2011-11-01	09:04:55.691	64.48	0.33	3.8	83.33	21.91	-0.5	-59.24	130.41	Normal Day
3004	2011-11-01	09:05:45.345	63.44	0.3	3.14	56.03	17.84	2.89	35.7	89.62	Normal Day
3004	2011-11-01	09:05:49.151	64.11	0.32	3.74	66.34	17.74	2.08	-26.18	111.41	Normal Day
3004	2011-11-01	10:03:37.494	64.17	0.32	3.6	79.46	22.05	-0.83	-60.47	123.01	Normal Day
3004	2011-11-01	10:04:06.666	63.04	0.3	2.67	56.85	21.31	0.7	5.84	58.97	Normal Day
3004	2011-11-01	10:34:14.397	61.98	0.32	2.67	57.24	21.46	-1.48	-8.35	104.74	Normal Day
3004	2011-11-01	10:36:39.086	62.63	0.39	2.4	50.64	21.07	0.93	7.73	57.25	Normal Day
3004	2011-11-01	10:53:18.774	62.83	0.31	2.47	48.62	19.7	-2.44	43.55	92.48	Normal Day
3004	2011-11-01	10:55:21.447	63.97	0.3	2.94	62.9	21.43	2.1	-5.5	61.96	Normal Day
3004	2011-11-01	10:57:03.756	63.64	0.31	4.74	93.73	19.77	-0.41	-58.61	126.02	Normal Day
3004	2011-11-01	10:57:53.442	61.22	1.56	7.01	91.93	13.12	2.68	13.83	91.4	Normal Day
3004	2011-11-01	10:58:36.747	63.65	0.33	3.6	86.17	23.91	-0.36	-58.15	122.5	Normal Day

Figure 16: Sample Relation -- AlertDB <-> GaitDB page with no option and no date chosen.

Now, if we have selected the user, didn't select any date, selected any of the 4 options and then we have clicked the search button, then this message, in Fig-17, will appear:

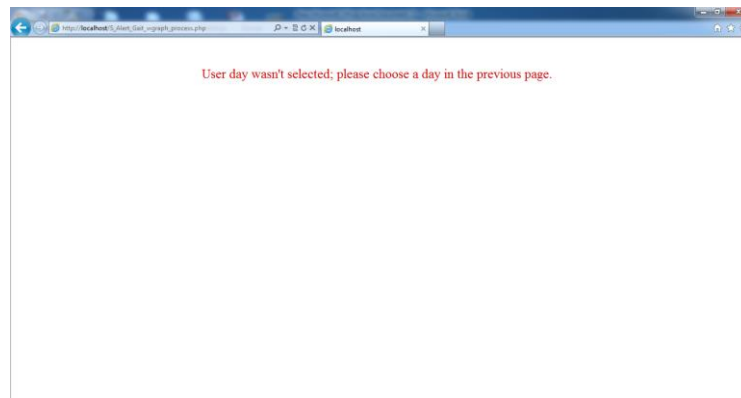


Figure 17: Relation -- AlertDB <-> GaitDB page with no date and any option chosen.

This message appears because to get the graph, we need to select a date first. The graph plots the *Speed/Duration/Length/Height* vs *System\_time* for that date.

### 5.1.5 The Relation -- AlertDB <-> GaitDB (Outliers detection) page

This page shows the data (Speed, Duration, Length) graphically. The graph is drawn like this: along the y-axis there is *Speed/Duration/Length* data and along the x-axis there is *Height* data. On this page we can view the outliers on the graph and on the data table as well. Outliers are the red colored points on the graph and red colored data on the data table. The outliers depend on the *Distance* value and *outliers day* we choose from *Days* dropdown list. The link for this page is: [http://localhost/s\\_alert\\_gait\\_wgraph\\_new1\\_allpara.php](http://localhost/s_alert_gait_wgraph_new1_allpara.php). The sample outlook of this page is depicted in Fig-18:

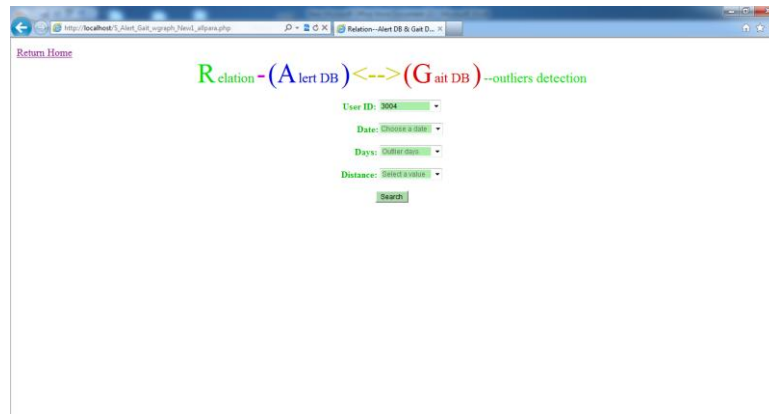


Figure 18: Outlook of the *Relation -- AlertDB <-> GaitDB (Outliers detection)* page.

Here, I filtered out the outliers from the actual residents on the basis of *Height* value. I took a window of 7 days (by selecting -7 from the *Day* dropdown list) from the day of interest (the selected day from the *Date* dropdown box) and calculated the Mean (Average) and Standard Deviation of *Height* value for that days (including the day of interest) and then calculated the outlier by the Equation.1 which is mentioned in Section 4.2.1. We can choose this *Distance* value from the 3rd dropdown box. We can test with different *Distance* value and see which *Distance* value is appropriate to show the outliers. I have put the values from 1 to 5 in the dropdown list to test which one



does the outliers detection for the users. I found out that for the *Distance* of 1, the outliers are filtered out perfectly; if we increase it to 2 or 3 or 4, then for some users, the outliers aren't detected properly. So, *Distance* value 1 was better for all the users. Now, to see the outliers detection graphically, we took UserID: 3050, Date: 2012-01-13, Days: -7 and Distance:1, we get the following output in Fig-19:

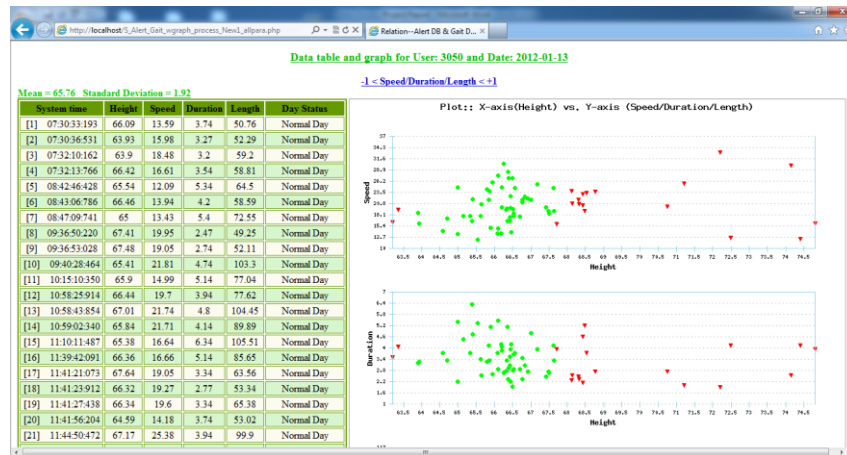


Figure 19: Outlier detection for the user data with *Distance* value 1

In Fig-19, on the graph the red values are the outliers and green values are the values which we will keep. The x-axis is *Height* and y-axis is *Speed*, *Duration* and *Length*. We can also check with the *Distance* value of 2 and that gives the output as in Fig-20:

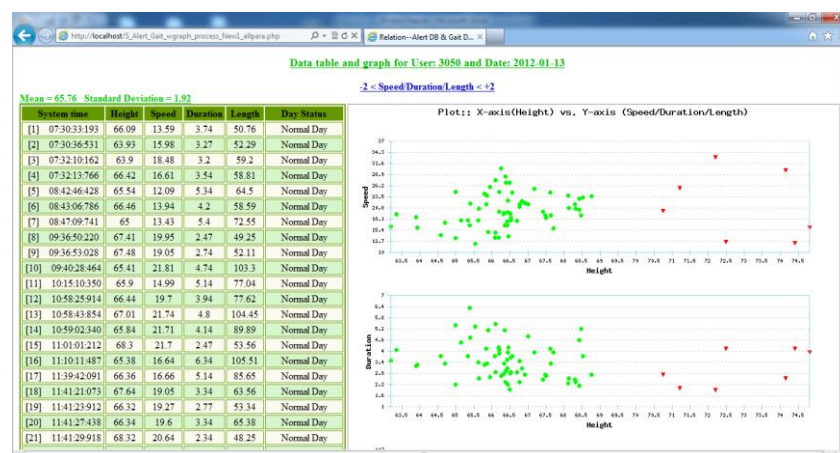
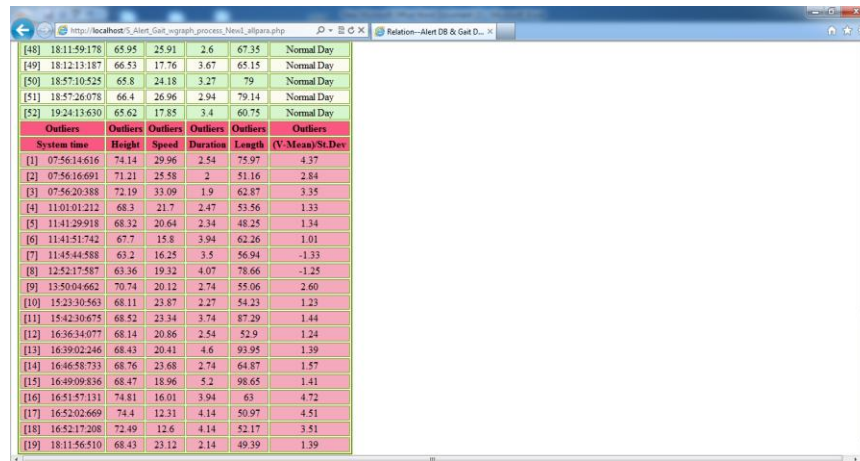


Figure 20: Outlier detection for the user data with *Distance* value 2



In Fig-20, we can see that the number of outliers is reduced because we choose a bigger distance. In Fig-19 and Fig-20, we can see the **green colored text** just above the data table, which shows the Mean (Average) and Standard deviation values for the 8 days. In Fig-20, we can see the **blue colored text** above the graph, which shows  $-2 < \text{Speed/Duration/Length} < +2$ . It means we selected the *Distance* 2, so the values outside the range of -2 to +2 will be detected as outliers. If we scroll down the page, then we will see the outliers at the bottom of the data table. These values are shown in red colors in the graph. Here is a snapshot of the outliers in the data table as in Fig-21:



	System time	Height	Speed	Duration	Length	(V-Mean)/Std. Dev
[48]	18.11.59.178	65.95	25.91	2.6	67.35	Normal Day
[49]	18.12.13.187	66.53	17.76	3.67	65.15	Normal Day
[50]	18.57.10.525	65.8	24.15	3.27	79	Normal Day
[51]	18.47.26.078	66.4	26.96	2.94	79.14	Normal Day
[52]	19.24.13.630	65.62	17.85	3.4	60.75	Normal Day
Outliers	Outliers	Outliers	Outliers	Outliers	Outliers	Outliers
[1]	07.56.14.616	74.14	29.96	2.54	75.97	4.37
[2]	07.56.16.691	71.21	25.58	2	51.16	2.84
[3]	07.56.20.388	72.19	33.09	1.9	62.87	3.35
[4]	11.01.01.212	68.3	21.7	2.47	53.56	1.33
[5]	11.41.29.918	68.32	20.64	2.34	48.25	1.34
[6]	11.41.51.742	67.7	15.8	3.94	62.26	1.01
[7]	11.45.44.588	63.2	16.25	3.5	56.94	-1.33
[8]	12.52.17.587	63.36	19.32	4.07	78.66	-1.25
[9]	13.50.04.662	70.74	20.12	2.74	55.06	2.60
[10]	15.23.30.563	68.11	23.87	2.27	54.23	1.23
[11]	15.42.30.675	68.52	23.34	3.74	87.29	1.44
[12]	16.36.34.077	68.14	20.86	2.54	52.9	1.24
[13]	16.39.02.246	68.43	20.41	4.6	93.95	1.39
[14]	16.46.58.733	68.76	23.68	2.74	64.87	1.57
[15]	16.49.09.836	68.47	18.96	5.2	98.65	1.41
[16]	16.51.57.131	74.81	16.01	3.94	63	4.72
[17]	16.52.02.669	74.4	12.31	4.14	50.97	4.51
[18]	16.52.17.208	72.49	12.6	4.14	52.17	3.51
[19]	18.11.56.510	68.43	23.12	2.14	49.39	1.39

Figure 21: Outlier in the data table for the user data with *Distance* value 1.

In Fig-21, if we look at the last column of the outliers data, we can see the formula “(V-Mean)/Std. Dev” which is same as Equation.1 of section 4.2.1. The values in this column are greater than +1 or less than -1; since we choose *Distance* value of 1, these values are greater than +1 or less than -1, hence these are outliers.

### 5.1.6 The Analysis -- AlertDB <-> GaitDB (With outliers) page

This page shows the Min, Max, Avg, Max-Avg and Avg-Min values for each day for a user. These values can be used for analyzing the parameter that shows the distinctive behavior for *Abnormal* day. Outliers are included in the results of this page. So, we don't use this page much. We use it just to see how the different values (Min , Max, Avg etc) were when we included the Outliers. The link for this page is: [http://localhost/s\\_alert\\_gait.php](http://localhost/s_alert_gait.php). We can compare the results from this page with the analysis page which shows the output without the outliers. Here is a snapshot of the front page of this page in Fig-22:

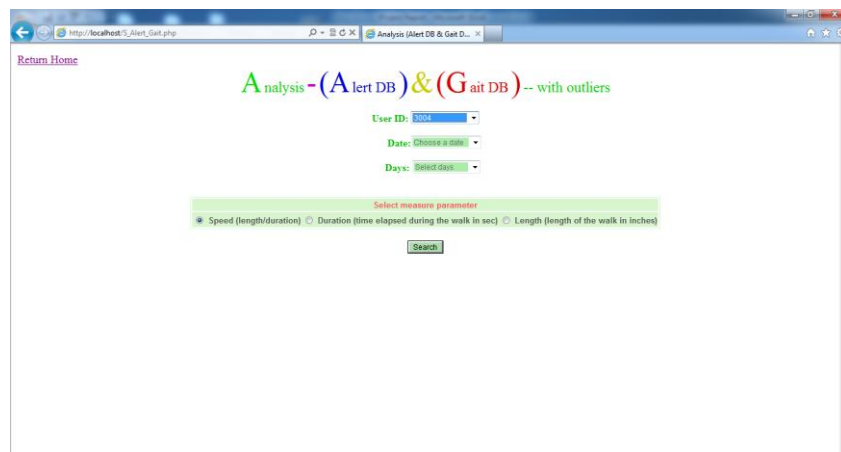


Figure 22: Sample outlook of the *Analysis -- AlertDB <-> GaitDB (With outliers)* page.

After selecting the *UserID* and *Date* from the 1<sup>st</sup> and 2<sup>nd</sup> dropdown lists, we can choose the number of previous days from the *Days* dropdown box (see the 3rd dropdown box in Fig-22). If we select a specific date from the *Date* dropdown box and then choose previous any number of days (5 or 10 or 12 days) (see the different numbers in the *Days* dropdown box with ‘-’ sign in front of them; negative means previous days), then we will be able to see the analysis result of those number of days. Let's see the output screen in Fig-23 when we selected *UserID* 3004, *Date* 2011-12-10 (which is an *Abnormal Day* for user 3004) and *Days* -14:

Result for user 3004 for Day:2011-12-10 and previous 14 days

Speed Data for Day (not divided into morning and evening session)

Min	Max	Avg	Max-Avg	Avg-Min	Date	Day status
13.69	27.08	21.36	5.72	7.67	2011-11-26	Normal Day
15.05	47.60	22.38	25.22	7.33	2011-11-27	Normal Day
15.97	28.56	20.77	7.79	4.80	2011-11-28	Normal Day
8.96	32.18	19.43	12.75	10.47	2011-11-29	Normal Day
10.99	24.87	18.25	6.62	7.26	2011-11-30	Normal Day
11.89	24.24	18.11	6.13	6.22	2011-12-01	Normal Day
11.06	24.72	20.54	4.18	9.48	2011-12-02	Normal Day
12.93	33.88	19.95	13.93	7.02	2011-12-03	Normal Day
13.56	26.55	20.04	6.51	6.48	2011-12-04	Normal Day
12.24	28.17	19.57	8.60	7.33	2011-12-05	Normal Day
11.09	29.49	19.87	9.62	8.78	2011-12-06	Normal Day
11.88	27.34	18.46	8.88	6.58	2011-12-07	Normal Day
12.34	21.19	16.38	4.81	4.04	2011-12-08	Normal Day
9.94	22.23	17.38	4.85	7.44	2011-12-09	Normal Day
11.22	32.42	19.16	13.26	7.94	2011-12-10	Abnormal Day

Speed Data for Morning session (07:00:00 to 12:59:59)

Figure 23: Sample output of the *Analysis -- AlertDB <-> GaitDB (With outliers)* page.

In Fig-23, we can see the previous 14 days results and the date we selected from the *Date* dropdown box is the last row of the table, which is an Abnormal day. The output shows the Min, Max, Avg etc. parameters in 3 tables: first table is for the whole day, second table is for the morning session (07:00:00 to 12:59:59) and third table is for the evening session (19:00:00 to 23:59:59).

#### 5.1.7 The *Analysis -- AlertDB <-> GaitDB (Without outliers)* page

This page shows the Min, Max, Avg, Max-Avg and Avg-Min values for each day for a user. These values can be used for analyzing the parameter that shows the distinctive behavior for Abnormal day. We can see up to 30 previous days' data for analyzing the Min, Max, Avg etc values. Outliers are NOT included in the results of this page. So, we use this page for the analysis of different parameters. This is the link for this page: [http://localhost/s\\_alert\\_gait\\_new\\_allpara.php](http://localhost/s_alert_gait_new_allpara.php).

This page shows the analysis results without the outliers. We can compare this page's result with the previous page of section 5.1.6 (which included outliers' results) and we can see that different parameter values are changed, They are changed because the outliers are excluded now, so the result only reflects the original residents, not the

outliers or visitors. We can see the any number of previous day's results and compare it with the parameters of the Abnormal Days and see how the different parameters change. I was informed to compare the parameters of Abnormal Day with previous 14 Normal Days' parameters. Here is a snapshot of the front page of this page in Fig-24:

Return Home

Analysis -- (Alert DB) & (Gait DB) --without outliers

User ID: 3004

Date: Choose a date

Days: Outlier days

Days: Compare days

Distance: Select a value

Search

Figure 24: Sample outlook of the *Analysis -- AlertDB <-> GaitDB (Without outliers)* page.

Let's say, we select -14 days from the *Days* dropdown box (compare days), -7 days from the outliers day dropdown box and *Distance:1* from the *Distance* dropdown box and then click search, we will see the following result as in Fig-25:

Result for user 3004 for Abnormal Day:2011-12-10 and previous 14 days

Speed/Duration/Length <+1

Analysis data for Speed parameter

Speed Data for Day (not divided into morning and evening session)

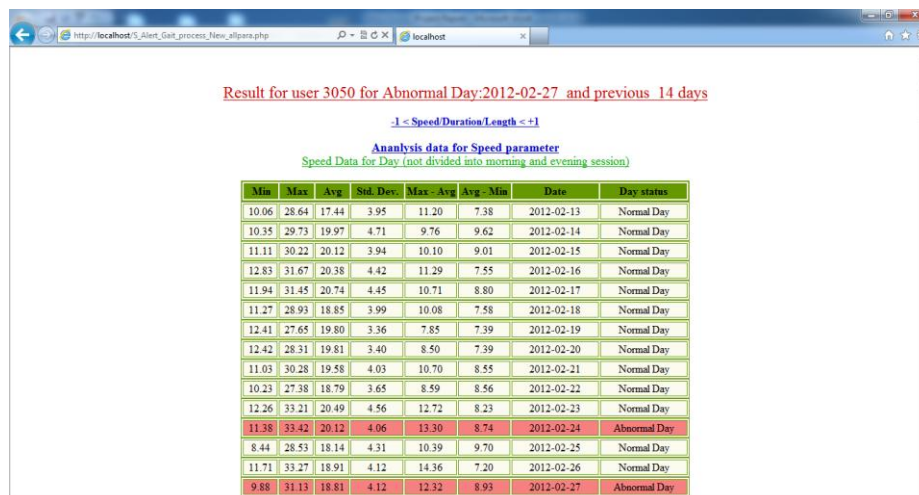
Min	Max	Avg	Std. Dev.	Max - Avg	Avg - Min	Date	Day status
13.69	27.08	21.37	2.65	5.71	7.68	2011-11-26	Normal Day
15.05	47.60	22.30	5.11	25.30	7.25	2011-11-27	Normal Day
16.15	28.56	21.22	2.67	7.34	5.07	2011-11-28	Normal Day
8.96	31.26	19.89	3.69	11.37	10.93	2011-11-29	Normal Day
10.99	24.87	18.57	3.35	6.30	7.58	2011-11-30	Normal Day
12.13	24.24	18.56	2.95	5.68	6.43	2011-12-01	Normal Day
15.91	24.72	21.01	1.82	3.71	5.10	2011-12-02	Normal Day
14.12	28.07	19.67	2.69	8.40	5.55	2011-12-03	Normal Day
13.56	26.55	20.02	3.02	6.53	6.46	2011-12-04	Normal Day
12.24	24.86	19.76	3.02	5.10	7.52	2011-12-05	Normal Day
11.09	29.49	20.09	3.97	9.40	9.00	2011-12-06	Normal Day
11.88	25.76	18.58	2.63	7.18	6.70	2011-12-07	Normal Day
12.94	21.19	17.10	2.11	4.09	4.16	2011-12-08	Normal Day
12.89	22.23	17.90	2.50	4.33	5.01	2011-12-09	Normal Day
11.99	32.42	18.62	3.61	13.79	6.64	2011-12-10	Abnormal Day

Speed Data for Morning session (07:00:00 to 12:59:59)

Figure 25: Sample output-1 of the *Analysis -- AlertDB <-> GaitDB (Without outliers)* page.

In Fig-25, we can see the previous 14 days results and the date we selected from the *Date* dropdown box is the last row of the table, which is an Abnormal Day.

I have provided the option to select not only -14 days but also any number of days from -1 to -30. The reason is, sometimes there are Abnormal Days within this previous 14 days; in such case select that number of Days from the dropdown box. Suppose, for User 3050, Date: 2012-02-27 is an Abnormal Day and 2012-02-24 is another Abnormal Day. So, if we select 2012-02-27 in the Date dropdown box, we should select -2 in the Days dropdown box. Because if we select -14 in this case, then the other Abnormal Day's data will be included and then the analysis result will be flawed. Fig-26 shows such an example case:



Result for user 3050 for Abnormal Day:2012-02-27 and previous 14 days

-1 < Speed/Duration/Length < +1

Analysis data for Speed parameter

Speed Data for Day (not divided into morning and evening session)

Min	Max	Avg	Std. Dev.	Max - Avg	Avg - Min	Date	Day status
10.06	28.64	17.44	3.95	11.20	7.38	2012-02-13	Normal Day
10.35	29.73	19.97	4.71	9.76	9.62	2012-02-14	Normal Day
11.11	30.22	20.12	3.94	10.10	9.01	2012-02-15	Normal Day
12.83	31.67	20.38	4.42	11.29	7.55	2012-02-16	Normal Day
11.94	31.45	20.74	4.45	10.71	8.80	2012-02-17	Normal Day
11.27	28.93	18.85	3.99	10.08	7.58	2012-02-18	Normal Day
12.41	27.65	19.80	3.36	7.85	7.39	2012-02-19	Normal Day
12.42	28.31	19.81	3.40	8.50	7.39	2012-02-20	Normal Day
11.03	30.28	19.58	4.03	10.70	8.55	2012-02-21	Normal Day
10.23	27.38	18.79	3.65	8.59	8.56	2012-02-22	Normal Day
12.26	33.21	20.49	4.56	12.72	8.23	2012-02-23	Normal Day
11.38	33.42	20.12	4.06	13.30	8.74	2012-02-24	Abnormal Day
8.44	28.53	18.14	4.31	10.39	9.70	2012-02-25	Normal Day
11.71	33.27	18.91	4.12	14.36	7.20	2012-02-26	Normal Day
9.88	31.13	18.81	4.12	12.32	8.93	2012-02-27	Abnormal Day

Figure 26: Sample output-2 of the *Analysis -- AlertDB <-> GaitDB (Without outliers)* page.

In Fig-26, we selected the 2012-02-27 date as Abnormal Date and selected -14 days and we can see that in the result, there is another Abnormal Day within -14 days. So, we should have selected -2 days because there are two normal days before this 2012-02-27 Abnormal Day. So in such case, we should take a look at the -14 days and if there are any Abnormal Days within this -14 Days then we should go back and select appropriate number of Days so that there is no Abnormal Day in between.

### 5.1.8 The Analysis -- AlertDB <-> GaitDB (Without outliers) (w/ graph) page

This page shows Min, Max, Avg. etc for the users which have both Abnormal and Normal days. The data for *whole day*, *morning session* and *evening session* of each user day are graphically shown for detecting the distinctive parameter for Abnormal Day. We can see up to 14 previous days' data in order to compare them with the currently selected Abnormal Day's data. This is the link for this page: [http://localhost/s\\_alert\\_gait\\_new1\\_allpara.php](http://localhost/s_alert_gait_new1_allpara.php).

This page shows the analysis results, without the outliers, graphically. We can easily view the changes of different parameters from the graph of this page and then decide which parameter (if there is any) can be used to show the Abnormal Day's effect on that parameter. Here is a snapshot of the first page of that page in Fig-27:

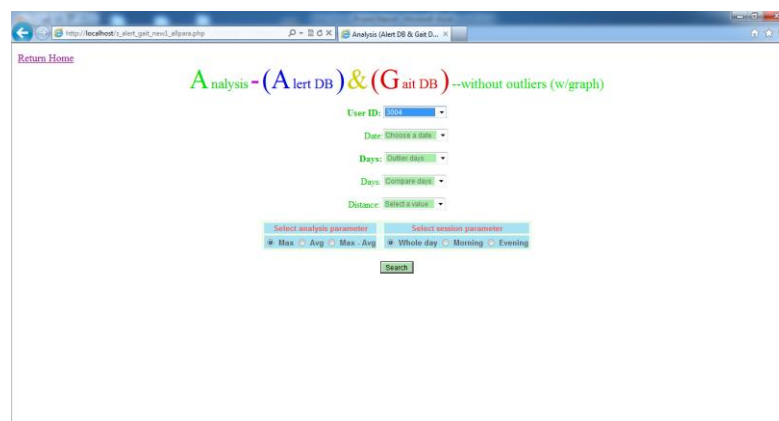


Figure 27: Outlook of Analysis -- AlertDB <-> GaitDB (Without outliers) (w/ graph) page.

In Fig-27, we can see 2 extra options in blue colors. For example, we want to see how the Abnormal Day: 2012-02-27 of user 3050's *Speed* parameter changes in comparing with previous 14 days, so we need to select that Abnormal Date and -14 Days, select *Max* from the 2nd Option box and *Whole Day* from the 3rd Option box and then press search.

*Note:* On the *UserID* dropdown box, I have only shown those users, which are single resident have both Normal and Abnormal Days in their entries. So I found out that

user 3004, 3013, 3037, 3038 and 3050 are single resident and they have both Normal and Abnormal Days. We can check from GaitDB that the other users either don't have Abnormal Days or they are double resident case (data of user 3017, 3047 and 3049 are with double resident data, so we didn't include them). In the *Date* dropdown box, I have only shown the Abnormal Day(s) for the user, so that we don't have to go through all the dates and select the Abnormal Day. Hence in order to analyze the data, we just need to select an user and an Abnormal Day, outliers day from Days dropdown box and appropriate number of previous day and then select distance (distance value of 1 is good for all the users) and choose the choice of session of the Day and then we can see the analysis result graphically along with the data table. Here is a sample snapshot, in Fig-28, of the output of this analysis page:

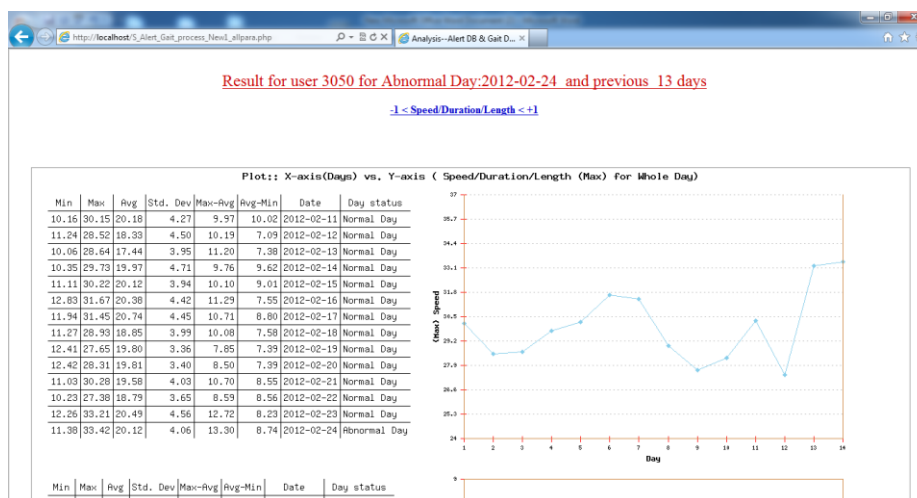


Figure 28: Output of the *Analysis -- AlertDB <-> GaitDB (Without outliers) (w/ graph)* page.

In Fig-28, we can see the previous 12 days results on the graph and on the data table, for the *Max* value of *Speed*, *Duration* and *Length* for the *Whole Day* session. The Abnormal date we selected from the *Date* dropdown box is the last row of the table and the rightmost point on the graph.



### 5.1.9 The Max detection -- AlertDB <-> GaitDB (All users & All Day) page

This page shows *Max* value and occurring time of *Max* value of *Speed*, *Duration* and *Length* parameter for all users and all days. In this page, different data are graphically shown: along y-axis: *Speed/Duration/Length* and x-axis: *System time*; outliers are filtered out. The link for this page is: [http://localhost/S\\_Alert\\_Gait\\_wgraph\\_max\\_allpara.php](http://localhost/S_Alert_Gait_wgraph_max_allpara.php). After selecting a *UserID*, *Date*, *Days* and *Distance* value, the resulting page will show like this in Fig-29:



Figure 29: Output of the *Max detection -- AlertDB <-> GaitDB (All users & All Day) page*.

In Fig-29, we can see the red colored text just above the data table. This shows the *Max* value of *Speed*, *Duration* and *Length* for the *Whole day* and the time at which the *Max* occurred. We can choose to see *Max* value for *Whole Day*, *Morning Session* or *Evening Session*. We need to select the right option from the first page.

### 5.1.10 The Max detection--AlertDB<->GaitDB(selected users & Abnormal Day) page

This page shows *Max* value and occurring time of *Max* value of *Speed*, *Duration* and *Length* parameter for the users which have both Abnormal and Normal days. In this page, different data are graphically shown: along y-axis: *Speed/Duration/Length* and



x-axis: *System time*; outliers are filtered out. The *UserID* dropdown list contains only the users which have both normal and abnormal days and the *Date* dropdown list contains only abnormal days for the selected user. The link for this page is: [http://localhost/S\\_Alert\\_Gait\\_wgraph\\_max1\\_allpara.php](http://localhost/S_Alert_Gait_wgraph_max1_allpara.php). The output from this page will be similar to Fig-29.

## **5.2 Data analysis result**

I analyzed the data in two phases. At first, I received Alert and Gait data which was from October 2011 to February 29, 2012. Later, I received data from March 01 2012 to May 14 2012. Hence I will discuss about the outcome of data analysis for these two phases separately.

### ***5.2.1 First Phase: for the data from October 2011 to February 29, 2012***

I tested several strategies to analyze the Abnormal days and Normal days for a certain user and I tried to detect which parameter(s) can be considered as distinctive by which the Abnormal or Normal days can be separated.

Among the 10 users, I found that only 3 users (3004, 3037 and 3050) are single residents and they have both Abnormal Days and Normal days. Besides, I only considered the dates, which are greater than 2011-10-31. Hence my initial considerations were:

- i) Take only those users who are single residents and which have both Abnormal and Normal days' entry
- ii) Consider the entries which have dates greater than 2011-10-31

After applying these conditions I found that User 3004, 3037 and 3050 fulfill our criteria. I will discuss about the strategies which I used to identify the parameter, which changes their values to reflect a day's status: Normal or Abnormal.

### 5.2.1.1 Approach -1: Minimum, Maximum and Average Speed:

At first, I calculated the *Minimum, Maximum, Average* and *Standard Deviation* of *Speed* parameter for a certain user for all the available dates. Then, I calculated the difference between the ‘average speed and maximum speed’ and the ‘average speed and minimum speed’ for that user for all the available dates. I repeated this process for all the 3 users and here is the outcome of such operation. I have shown here the data of Abnormal Day and previous 14 Normal Days (for 3004 and 3037) and previous 13 days for one of the Abnormal days of user 3050 (3050 has 3 Abnormal days).

#### User: 3004

Min	Max	Avg	Std. Dev.	Max - Avg	Avg - Min	Date	Day status
13.69	27.08	21.37	2.65	5.71	7.68	2011-11-26	Normal Day
15.05	47.60	22.30	5.11	25.30	7.25	2011-11-27	Normal Day
16.15	28.56	21.22	2.67	7.34	5.07	2011-11-28	Normal Day
8.96	31.26	19.89	3.69	11.37	10.93	2011-11-29	Normal Day
10.99	24.87	18.57	3.35	6.30	7.58	2011-11-30	Normal Day
12.13	24.24	18.56	2.95	5.68	6.43	2011-12-01	Normal Day
15.91	24.72	21.01	1.82	3.71	5.10	2011-12-02	Normal Day
14.12	28.07	19.67	2.69	8.40	5.55	2011-12-03	Normal Day
13.56	26.55	20.02	3.02	6.53	6.46	2011-12-04	Normal Day
12.24	24.86	19.76	3.02	5.10	7.52	2011-12-05	Normal Day
11.09	29.49	20.09	3.97	9.40	9.00	2011-12-06	Normal Day
11.88	25.76	18.58	2.63	7.18	6.70	2011-12-07	Normal Day
12.94	21.19	17.10	2.11	4.09	4.16	2011-12-08	Normal Day
12.89	22.23	17.90	2.50	4.33	5.01	2011-12-09	Normal Day
11.99	32.42	18.62	3.61	13.79	6.64	2011-12-10	Abnormal Day

#### User: 3050

Min	Max	Avg	Std. Dev.	Max - Avg	Avg - Min	Date	Day status
10.16	30.15	20.18	4.27	9.97	10.02	2012-02-11	Normal Day
11.24	28.52	18.33	4.50	10.19	7.09	2012-02-12	Normal Day
10.06	28.64	17.44	3.95	11.20	7.38	2012-02-13	Normal Day
10.35	29.73	19.97	4.71	9.76	9.62	2012-02-14	Normal Day
11.11	30.22	20.12	3.94	10.10	9.01	2012-02-15	Normal Day
12.83	31.67	20.38	4.42	11.29	7.55	2012-02-16	Normal Day
11.94	31.45	20.74	4.45	10.71	8.80	2012-02-17	Normal Day
11.27	28.93	18.85	3.99	10.08	7.58	2012-02-18	Normal Day
12.41	27.65	19.80	3.36	7.85	7.39	2012-02-19	Normal Day
12.42	28.31	19.81	3.40	8.50	7.39	2012-02-20	Normal Day
11.03	30.28	19.58	4.03	10.70	8.55	2012-02-21	Normal Day
10.23	27.38	18.79	3.65	8.59	8.56	2012-02-22	Normal Day
12.26	33.21	20.49	4.56	12.72	8.23	2012-02-23	Normal Day
11.38	33.42	20.12	4.06	13.30	8.74	2012-02-24	Abnormal Day

#### User: 3037

Min	Max	Avg	Std. Dev.	Max - Avg	Avg - Min	Date	Day status
10.44	28.81	17.31	4.37	11.50	6.87	2011-12-31	Normal Day
10.98	23.82	16.04	2.64	7.78	5.06	2012-01-01	Normal Day
11.33	28.06	16.80	3.07	11.26	5.47	2012-01-02	Normal Day
10.02	33.36	17.63	4.55	15.73	7.61	2012-01-03	Normal Day
12.82	30.00	18.93	4.26	11.07	6.11	2012-01-04	Normal Day
9.60	22.45	16.54	2.72	5.91	6.94	2012-01-05	Normal Day
9.75	24.00	16.35	2.92	7.65	6.60	2012-01-06	Normal Day
10.85	34.60	16.66	4.34	17.94	5.81	2012-01-07	Normal Day
11.33	21.42	15.91	2.30	5.51	4.58	2012-01-08	Normal Day
12.12	30.00	17.25	3.75	12.75	5.13	2012-01-09	Normal Day
11.00	25.21	15.77	2.67	9.44	4.77	2012-01-10	Normal Day
11.76	30.74	18.89	4.76	11.85	7.13	2012-01-11	Normal Day
10.19	30.53	15.51	3.37	15.02	5.32	2012-01-12	Normal Day
11.69	27.74	17.55	3.05	10.19	5.86	2012-01-13	Abnormal Day

Now, I checked whether the **Max** value for a user shows *certain behavior* for **Abnormal** and **Normal** days. We can see from the above tables that, for **user 3004**,

- i) the **Max** value for **Abnormal day is: 32.42**, which is **neither the greatest nor the smallest value** than the **Max** values for the Normal days.
- ii) besides, the **Min** value for **Abnormal day is: 11.99**, which is also **neither the lowest nor the greatest value** than the **Min** values for the Normal days.
- iii) these also occur for the '**Max-Avg**' (max minus avg) and '**Avg-Min**' (avg minus min) values as well. **None of these indicate a solid identification for Abnormal and Normal days.**

For the **other users as well**, user 3050 and user 3037, I **did not find any distinctive feature in Max, Min, 'Avg-Min' or 'Max-Avg'** which can distinguish between Abnormal and Normal days.

So, I **ruled out** this approach and approached for a more **detailed method**.

#### 5.2.1.2 Approach-2: Session based Minimum, Maximum and Average Speed:

In this approach, I divided each of the user days into two sessions: Morning session and Evening session. I considered the **Morning session from 07:00:00 to 12:59:59**; and, the Evening session from **19:00:00 to 23:59:59**. By dividing the days into 2 sessions, I got the following results for the User: 3004.

**Speed Data for Morning session (07:00:00 to 12:59:59)**

Min	Max	Avg	Std. Dev.	Max - Avg	Avg - Min	Date	Day status
13.69	26.96	20.05	3.21	6.91	6.36	2011-11-26	Normal Day
18.94	25.62	22.52	2.15	3.10	3.58	2011-11-27	Normal Day
16.15	24.31	21.76	2.33	2.55	5.61	2011-11-28	Normal Day
18.66	31.26	21.96	3.08	9.30	3.30	2011-11-29	Normal Day
12.80	24.51	19.57	3.29	4.94	6.77	2011-11-30	Normal Day
12.13	24.24	18.90	3.43	5.34	6.77	2011-12-01	Normal Day
17.30	24.72	21.02	1.85	3.70	3.72	2011-12-02	Normal Day
15.57	23.69	19.95	2.47	3.74	4.38	2011-12-03	Normal Day
18.35	24.81	21.88	2.11	2.93	3.53	2011-12-04	Normal Day
12.75	24.86	20.57	2.94	4.29	7.82	2011-12-05	Normal Day
14.44	29.49	20.42	3.67	9.07	5.98	2011-12-06	Normal Day
11.88	22.97	18.57	2.48	4.40	6.69	2011-12-07	Normal Day
12.94	19.94	16.42	2.10	3.52	3.48	2011-12-08	Normal Day
14.08	22.23	18.18	2.63	4.05	4.10	2011-12-09	Normal Day
12.79	32.42	18.63	4.75	13.79	5.84	2011-12-10	Abnormal Day

**Speed Data for Evening session (19:00:00 to 23:59:59)**

Min	Max	Avg	Std. Dev.	Max - Avg	Avg - Min	Date	Day status
19.13	22.85	21.49	1.36	1.36	2.36	2011-11-26	Normal Day
15.63	22.85	20.23	2.13	2.62	4.60	2011-11-27	Normal Day
18.00	21.65	19.92	1.43	1.73	1.92	2011-11-28	Normal Day
15.06	22.00	18.70	2.19	3.30	3.64	2011-11-29	Normal Day
10.99	24.87	17.88	3.50	6.99	6.89	2011-11-30	Normal Day
12.33	22.21	17.50	3.22	4.71	5.17	2011-12-01	Normal Day
15.91	22.89	20.48	2.71	2.40	4.57	2011-12-02	Normal Day
16.63	20.66	18.15	1.64	2.51	1.52	2011-12-03	Normal Day
13.75	21.63	18.63	2.70	3.00	4.88	2011-12-04	Normal Day
15.81	21.88	18.66	2.18	3.22	2.85	2011-12-05	Normal Day
17.50	19.32	18.41	0.91	0.91	0.91	2011-12-06	Normal Day
15.81	16.18	16.00	0.18	0.18	0.18	2011-12-07	Normal Day
14.70	17.79	16.62	1.06	1.17	1.92	2011-12-08	Normal Day
12.89	20.72	16.84	2.49	3.88	3.95	2011-12-09	Normal Day
11.99	30.39	19.22	4.36	11.17	7.23	2011-12-10	Abnormal Day

Here, for the **Morning session**, Max and Max-Avg shows distinctive feature:

- i) The **Max** value for **Abnormal day** is the **greatest** (32.42) in comparing with **Max** values for the **Normal days**.
- ii) The **Max-Avg** value for **Abnormal day** is the **greatest** (13.79) in comparing with **Max-Avg** values for the **Normal days**.
- iii) The **Min**, and '**Avg-Min**' don't show any distinctive feature.

For the **Evening session**, we can see distinctive behavior for the **Max** and '**Max-Avg**' values.

- i) For the **Abnormal day**, the **Max** value is 30.39, which is the **highest Max** value in comparing with the **Max** values of **Normal days**.
- ii) The **Max-Avg** value for **Abnormal day** is the **greatest** (11.17) in comparing with **Max-Avg** values for the **Normal days**.
- iii) The **Avg-Min** value for **Abnormal day** is the **greatest** (7.23) in comparing with **Avg-Min** values for the **Normal days**.
- iv) The **Min** doesn't show any distinctive feature.

I analyzed user 3050's data similarly and found out that the **Max** and '**Max-Avg**' for the **Evening Session** are distinctive for both the users **3004** and **3050**. For User 3037, I found out that none of the parameters were distinctive.

Since, for User 3004 and 3050, I found out that Max and Max-Avg are distinctive, we can say that these two parameters can be used to identify the Abnormal Day from the Normal days. We need to see whether user 3037 is actually a single or double resident case. So, after analyzing the Min, Max, Avg-Min and Max-Avg values for the Abnormal days for these 3 users, we can come to the conclusion that,

- i) The **Min** and **Avg-Min** values **can be taken out of consideration** because they **don't show** representative feature for all the 3 users.

ii) The **Max** and '**Max-Avg**' shows distinctive feature for **user 3004 and 3050**, so we can consider these two parameters as the distinctive features, which can show the effect of Abnormal Day.

Therefore, we can say that these parameters **Max** and '**Max-Avg**' show some **distinctive** behavior for the **Speed** parameter for the users 3004 and 3050.

#### 5.2.1.3 Approach-3: Evening Session based Max and Avg-Max Duration:

Here, I analyzed the Max and Max-Avg values for *Duration* parameter of user 3004 and 3050 for the Evening session and found out that, they represent distinctive feature for this parameter. Here is the Evening session's data for user 3004 and 3050:

**User: 3004**

Min	Max	Avg	Std. Dev.	Max - Avg	Avg - Min	Date	Day status
2.40	4.00	3.00	0.59	1.00	0.60	2011-11-26	Normal Day
2.40	4.34	3.13	0.69	1.21	0.73	2011-11-27	Normal Day
2.34	2.94	2.65	0.27	0.29	0.31	2011-11-28	Normal Day
2.47	5.81	3.25	0.93	2.56	0.78	2011-11-29	Normal Day
2.54	5.54	3.58	0.78	1.96	1.04	2011-11-30	Normal Day
2.67	5.27	4.04	0.78	1.23	1.37	2011-12-01	Normal Day
2.20	5.54	3.12	1.40	2.42	0.92	2011-12-02	Normal Day
3.14	5.21	4.07	0.77	1.14	0.93	2011-12-03	Normal Day
2.27	5.41	3.50	1.07	1.91	1.23	2011-12-04	Normal Day
2.94	5.07	3.90	0.77	1.17	0.96	2011-12-05	Normal Day
3.07	6.28	4.68	1.61	1.61	1.61	2011-12-06	Normal Day
5.47	5.54	5.50	0.04	0.04	0.04	2011-12-07	Normal Day
3.60	6.01	4.49	0.96	1.52	0.89	2011-12-08	Normal Day
2.94	5.94	3.66	0.87	2.28	0.72	2011-12-09	Normal Day
1.80	6.34	3.51	0.95	2.83	1.71	2011-12-10	Abnormal Day

**User: 3050**

Min	Max	Avg	Std. Dev.	Max - Avg	Avg - Min	Date	Day status
2.54	3.34	2.96	0.28	0.38	0.42	2012-02-11	Normal Day
2.67	2.80	2.74	0.06	0.06	0.06	2012-02-13	Normal Day
2.74	5.54	3.93	0.93	1.61	1.19	2012-02-14	Normal Day
2.87	4.34	3.67	0.52	0.67	0.80	2012-02-15	Normal Day
2.74	4.47	3.70	0.62	0.76	0.96	2012-02-16	Normal Day
2.14	3.80	2.97	0.83	0.83	0.83	2012-02-17	Normal Day
3.20	3.47	3.34	0.13	0.13	0.13	2012-02-19	Normal Day
2.60	5.60	3.69	0.93	1.91	1.09	2012-02-21	Normal Day
3.20	4.54	4.00	0.58	0.54	0.80	2012-02-22	Normal Day
2.80	3.80	3.47	0.29	0.33	0.67	2012-02-23	Normal Day
2.27	5.94	3.50	0.81	2.44	1.23	2012-02-24	Abnormal Day

For these two users, the **Max** and **Max-Avg** parameters show distinctive behavior for the Abnormal day. The **Max** and **Max-Avg** both have highest Duration value for the Abnormal day in comparing with the Normal days. Hence, all the data of **Evening session** for user 3004 and 3050 for the parameters **Speed** and **Duration** support the hypothesis stated below. A day can be indicated as an Abnormal day, when,

- The **Max** value of **Speed/Duration** parameter of that day is *Highest* in comparing with the other **Max** values of Normal days, and
- The **Max-Avg** value of **Speed/Duration** parameter of that day is *Highest* in comparing with the other **Max-Avg** values of Normal days.

### Graphical representation:

Below is the graphical analysis of the **Max** and **Max-Avg** values for Evening session for the **Speed** and **Duration** parameter for the user 3004. For the Abnormal day (2011-12-10), the **Max** and **Max-Avg** has the highest value in comparing with other **Max** and **Max-Avg** values of the Normal days. The Abnormal day and previous 14 days' data's graphical plot is depicted below in Fig-30:

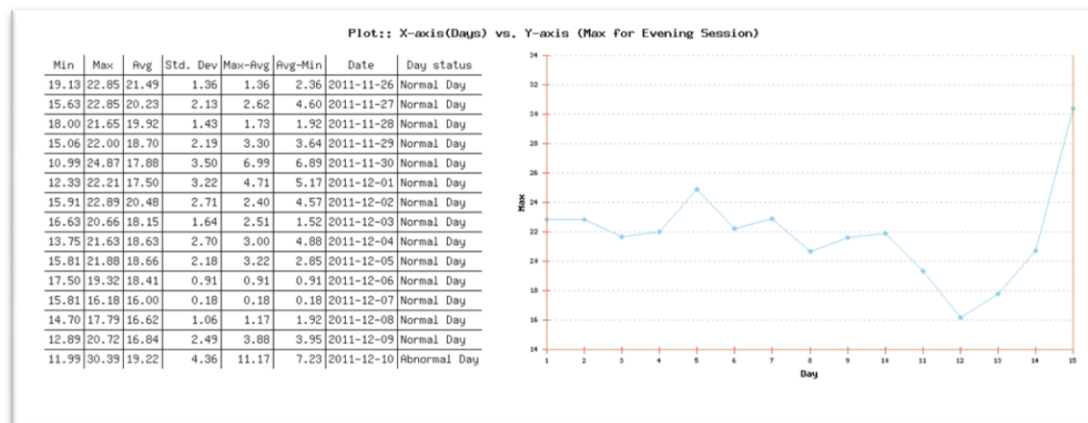


Figure 30: Data analysis output for Evening session data of Max Speed

Here, on the plot of Fig-30, Max of Abnormal day has the highest value; the rightmost point is the Abnormal day's value. Besides, there is a petty sharp raise in Speed for abnormal day in comparing with the previous 2 or 3 Normal days.

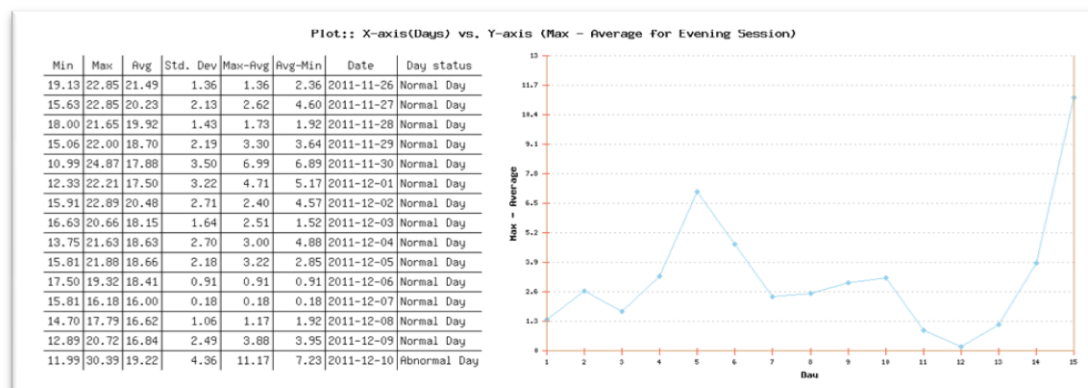


Figure 31: Data analysis output for Evening session data of Max-Avg Speed

Here as well, on the plot of Fig-31, Max-Avg of Abnormal day has the highest value; the rightmost point is the Abnormal day's value. Besides, there is a petty sharp raise in Speed for abnormal day in comparing with the previous 2 or 3 Normal days.

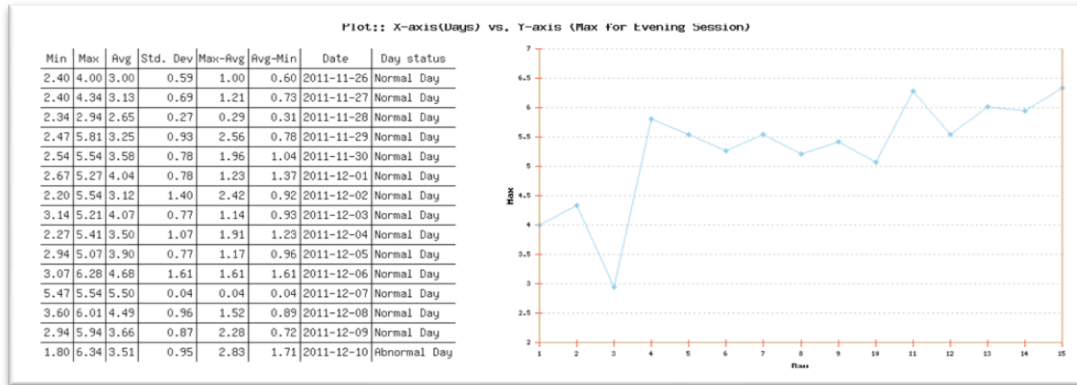


Figure 32: Data analysis output for Evening session data of Max Duration

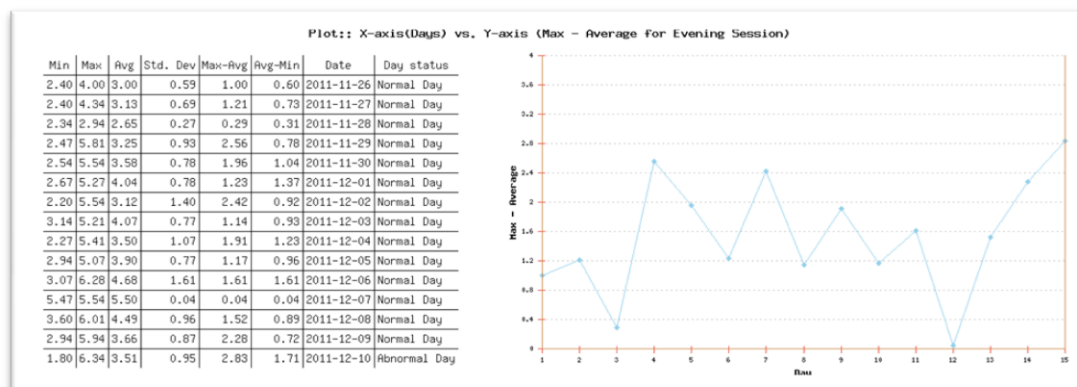


Figure 33: Data analysis output for Evening session data of Max-Avg Duration

Here as well, on the plot of Fig-32 and Fig-33, Max and Max-Avg of Abnormal day has the highest value; the rightmost point is the Abnormal day's value. Besides, there is a petty sharp raise in Max-Avg Duration for abnormal day in comparing with the previous 2 or 3 Normal days. I performed the similar plot analysis for the user 3050 and found out similar change in **Max** and **Max-Avg** parameters for the Speed and Duration data.

Therefore, after analyzing the tabular data and graphical data, we can say that the **Max** and **Max-Avg** can be used as **distinctive** parameter, which can show the changes for Abnormal Day.

### 5.2.2 Second Phase: for the data from October 2011 to May 14, 2012

In this phase, I worked with the new updates databases which contained data until May 14, 2012. I already had the data which was from October 2011 to February, 2012; the new data needed to be analyzed in the similar way I did for the old data.

With the old data, I had only 3 users which had both normal and abnormal day and single resident. Those users were: 3004, 3037 and 3050. I found out that **Max Speed** for **evening session** was distinctive parameter which can show certain behavior for Abnormal day.

Now, with the new data, I have total 5 users who have both Normal and Abnormal day and also single resident, those are: 3004, 3013, 3037, 3038 and 3050. So, let's apply the strategies to this new data which I applied to the previous data.

Let's see for 3004, if the new date(s) can show any distinctive parameter. I took user 3004 and date: 2012-03-05 with distance value of 1 and take 6 previous days and here is the result in Fig-34:

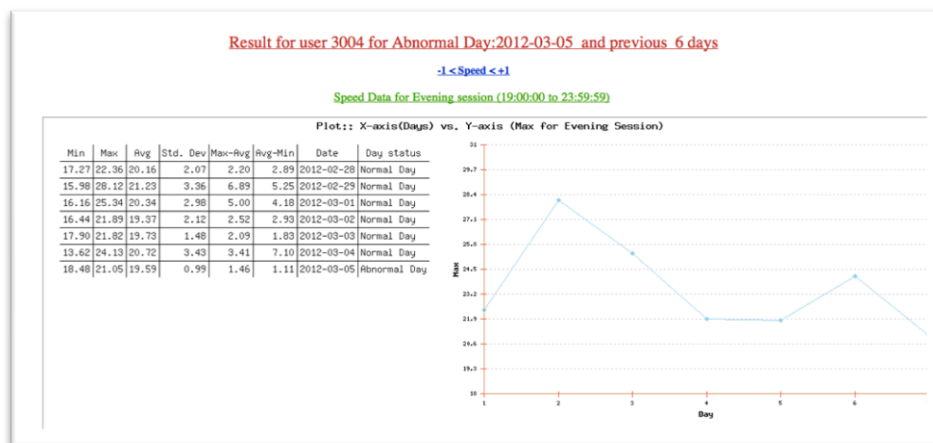


Figure 34: Data analysis output for Evening session data of Max Speed for new data



We can see that, the Max is no longer highest (in the Fig-34) as we can see it (in Fig-35) for date: 2011-12-10 (old data date):

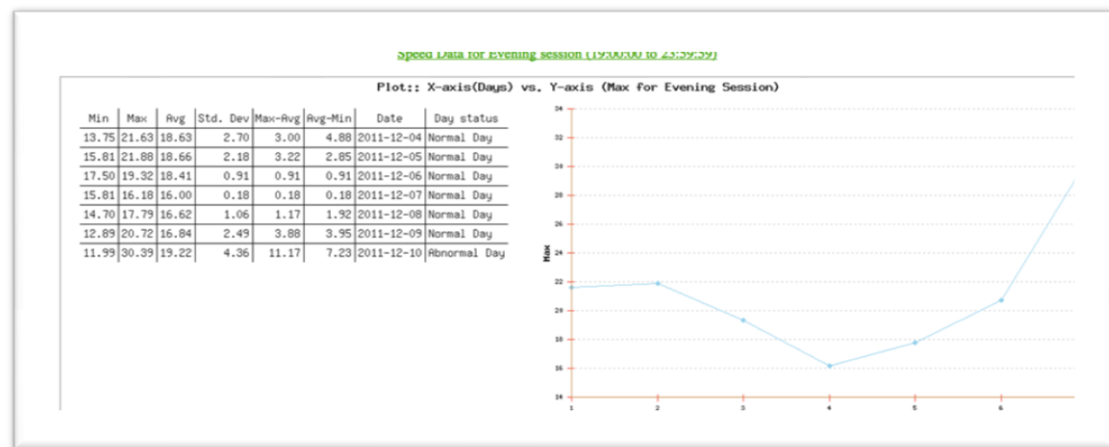


Figure 35: Data analysis output for Evening session data of Max Speed for old data

I tried with another date: 2012-04-28 for user 3004 and found that Max Speed is distinctive for this date, here is the output graph in Fig-36:

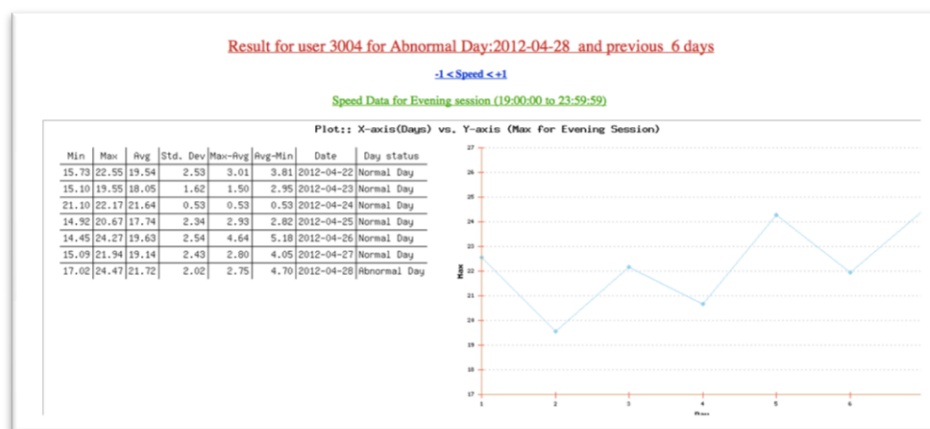


Figure 36: Data analysis output for Evening session data of Max Speed for new data

Here, in Fig-36, Max is highest for Abnormal day. However, for the other users, for new Abnormal date from new data, we did not find that Max is highest.

So, I thought that I would take two parameters and see if the combination of those two parameters can show any distinctive behavior for abnormal day.

I took Speed and Duration and tried to check if for an Abnormal user day,

- i) When Max Speed is highest, Max duration is also highest or Max Duration is lowest?
- ii) When Max Speed is lowest, Max duration is highest or Max Duration is highest?

But, I did not find any such behavior.

I also checked similar with *Speed* and *Length* parameter. But, there was also no such result. There was actually no such parameter which was showing certain behavior for Abnormal days for the new abnormal dates from new data.

Hence, I suppose, for the new data we cannot find any such distinctive parameter which shows certain attitude for Abnormal day.

## **6. Conclusion and Future Work**

The usage of technology such as non-wearable sensors, in the living places where the senior citizens prefer to live independently, can help early detection of the changes like reduced activity in the elder's apartment and can also help by firing alarm for the concerned personnel to intervene at proper time. A camera system together with this sensor-system provides much better clinical information to analyze the situation of the residents. These systems provide new ways of detecting subtle changes that do not require traditional face-to-face assessment of individual residents.

I was provided information about the resident's clinical history which were rated by the clinical experts; the experts analyzed the output from the sensors and then rated the status of the residents accordingly. The clinical information was provided to me in the form of a database named as AlertDB. The second database which I was provided was GaitDB, which contained information about the gait data of the residents gathered from the camera system deployed in the TigerPlace residence.

Now, the effectiveness of clinical information and gait information as a single healthcare system depends on the correct analysis of the obtained data from both the AlertDB and GaitDB. I was supplied these two DBs to classify the days of the TigerPlace residents into Normal/Abnormal category on the basis of the rating provided by the clinical experts. With this day status I have to identify from gait database which dates are Normal/Abnormal and which parameter of gait data can show the changes for the Abnormal day.

On the completion of the project, I successfully classified the AlertDB data into Normal and Abnormal user days based on the rating provided by the clinical experts. I also processed the gait data, which comes in text files, into correct format and assigned correct UserID for the files. After the alert data was classified and gait data were processed properly, I imported them into one common platform, phpMyAdmin MySQL platform, so that I can access them and try different operation on them from one single platform. Then, I made relation between these two databases on the basis of date and user-id. After these two databases were connected, I analyzed different attributes of gait data such as the speed of walk, duration of walk etc. by taking Min, Max, Avg. of them for the Normal and Abnormal Days. I created PHP pages to show the different results in order to analyze the AlertDB and GaitDB and also to visualize graphically different parameters' behavior for Normal and Abnormal days.

For the data dated from October, 2011 to February, 2012, I found that *Max* and *Max-Avg* values of Speed parameter shows a certain behavior (*Max Speed* was highest for Abnormal day than for the Normal days) for Abnormal day in comparing with Normal days. But, for the new data dated from March, 2012 to May 14, 2012, I could not find any such parameter which can show certain behavior for Abnormal day. It

might be that, when we will get more data in future, we can find a certain parameter which can show such distinctive behavior for the Abnormal day.

I have provided matlab codes by which the future gait data can be processed in correct format and imported into the combined database easily. I have also kept the opportunity in the PHP pages so that anyone can analyze the data statistically and visually and can identify such distinctive parameter, if there is any, which would show certain behavior for the Abnormal user day. By analyzing the future data, hopefully an effective clinical system can be developed by which the healthcare system, for the senior citizens who lives independently, will be benefited.

## **References:**

- [1]. Hayes TL, Pavel M, Kaye JA (2004). An unobtrusive in-home monitoring system for detection of key motor changes preceding cognitive decline. Proc. of the 26th Annual Intl. Conf. of the IEEE EMBS, pp. 2480-2483, San Francisco, CA.
- [2]. Mihail Popescu, George Chronis, Rohan Ohol, Marjorie Skubic, Marilyn Rantz (2011). An eldercare electronic health record system for predictive health assessment. 13th IEEE International Conference on e-Health Networking, Application, & Services.
- [3]. Gregory L. Alexander, Bonnie J. Wakefield, Marilyn Rantz, Marjorie Skubic, Myra A. Aud, Sanda Erdelez and Said Al Ghenaimi (2011). Passive sensor technology interface to assess elder activity in independent living. Nursing Research. September/October 2011 Vol 60, No 5, 318–325.
- [4]. Rantz, M.J., Skubic, M., Koopman, R.J., Alexander, G., Phillips, L., Musterman, K.I., Back, J.R., Aud, M.A., Galambos, C., Guevara, R.D., & Miller, S.J. (2012). Automated technology to speed recognition of signs of illness in older adults. Journal of Gerontological Nursing. Year: 2012, Vol. 38, No. 4, 18-23.
- [5]. Gregory L. Alexander, Marilyn Rantz, Marjorie Skubic, Richelle J. Koopman, Lorraine J. Phillips, Rainer D. Guevara, Steven J. Miller MA (2011). Evolution of an early illness warning system to monitor frail elders in independent living. Journal of Healthcare Engineering · Vol. 2 · No. 3 · 2011 Page 337–363.