Software Design Document: Capacity Monitoring Tool

Ala'a Mostafa, Dalia Ashry, Merna Osama, Nora Eleish
Supervised by
Dr. Abdul Rahman Galal Eng. Radwa Samy

June 18, 2017

1 Introduction

1.1 Purpose

The aim of the software design document is to fulfill the requirements of our proposed system that will be built according to these diagrams: class diagram, database diagram, use-case diagram, sequence diagram, data flow diagram, and activity diagram. Moreover, this document will illustrate the architecture of our proposed system.

1.2 Scope

Capacity monitoring tool aims to detect critical network loads through a web-based graphical user interface tool and a mobile application. The web-based keeps monitoring the network resources as the capacity of the nodes. Moreover, receiving alarms through an android mobile application when there is a critical capacity issue that is about to occur. The tool aims to serve Vodafone's capacity team in monitoring the capacity of the network nodes in order to prevent and fix this critical capacity issues that might come up. Operations will be performed on the data coming from the nodes such as data analysis, classification or clustering. In addition, providing dashboards that contain a summary of the data collected by the nodes through data visualization as in graphs, charts, and gauges.

1.3 Overview

This document is divided into 8 main components. The first section is about introduction discussing the document purpose and its scope.The second section is about system overview, discussing the idea of the proposed system in detail. The third section is about the architecture design of the proposed system, the data flow diagram, activity diagram and sequence diagram. The fourth section discusses database.The fifth section discusses class diagram.The sixth section displays system interface.The Seventh section views the requirement matrix.The eighth section is for references.

2 System Overview

The proposed tool consists of two parts: a website and an android mobile application.The website collects regularly different files with different formats as (JSON, XML, Excel,...etc.) from the following network nodes: Receiver Buffer Descriptor (RBD), Missed Call Keeper (MCK), Switch Divert(SD), and Session Description Protocol (SDP)...etc. As the website parses these collected files, convert the data into JSON format, and inserts the needed data into Firebase RealTime database. Once parsing, and converting operations are done, further operations can be performed on the data extracted from these files as data analysis, classification or clustering. Data Analysis includes the following operations: average, sum, maximum and minimum to produce reports.Dashboards are provided that display a summary of the analyzed data. The android mobile application will be developed for setting a maximum, minimum, and equal thresholds for a certain attribute. In case, a value exceeds a certain maximum threshold that was defined by the user earlier, an alarm will be sent and received via email, sms, or the application itself. In addition, the android mobile application will have a dashboard displaying a summary for the capacities of the nodes. Figures and analysis of data will be shown in the dashboard. The Call is Represented by:

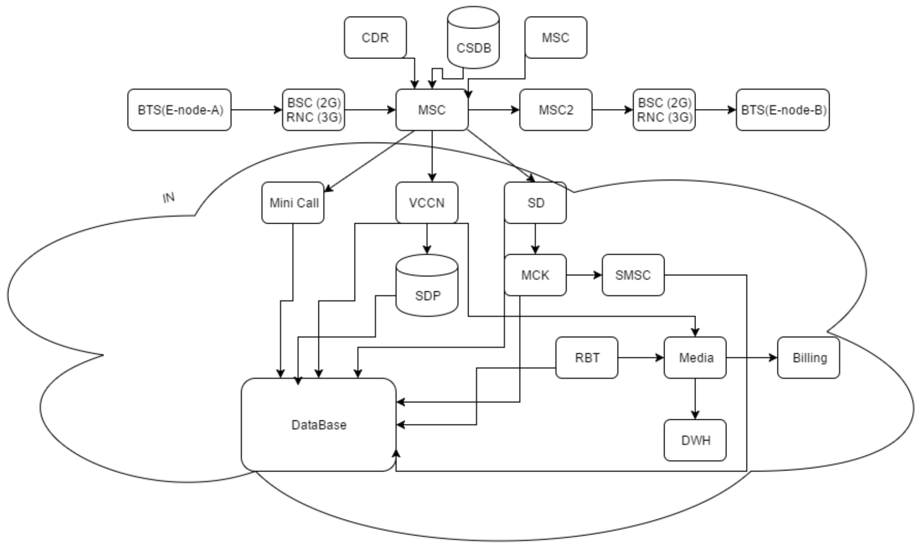


Figure 1: Vodafone Call Flow

Node A is the mobile device that wants to call node B which is another device but before reaching to node B, here are other nodes must be reached first, as first if it is 2G it must stop by Base Station Controller(BSC) node first, but if it is 3G it must stop by Radio Network Controller(RNC) node first ,but for 4G it doesn't have a specified node to stop by at the beginning it just goes through the Mobile Switching Center(MSC) node like the others also. Then through the Mobile Switching Center (MSC) node it search for the location of node B as Mobile Switching Center (MSC) is the core switching node it connects with the CSDB node which is the database that contains the location of every node and it found node B at Mobile Switching Center 2(MSC2) then it goes to Base Station Controller (BSC) node then node B which the phone rings.

The Mobile Switching Center (MSC) node contains the Intelligent Network (IN) which consists of the IT (Information Technology) department which is the core of development. In the Intelligent Network (IN) there is the Session Description Protocol (SDP) node which is the most secured node in Egypt as it contains all the data of the Vodafone customers.

In the Intelligent Network (IN), before the phone call reaches device B we must check if device A has enough balance/credit or not to make the call. VCCN is the charging node sent a request to Session Description Protocol (SDP) to complete the process if this user has credit to complete the call or not in each minute the VCCN sends a request to know the credit the user if the user has a credit the call will be continued, if not the call will end. All the reporting and log files will be exported from the Session Description Protocol / Service Data Point /Signal Distribution Panel (SDP).

The capacity monitoring tool system consists of three types of configurations which are data collection, system and job illustrate. The system, it consists of data like what are Internet Protocols (IPs) that will be reached to, what is the path, what is the username, what is the password and what is the file that will be reached through certain path These are the data which are important to be configured in the system, the system is not interested with what is inside the file. Data collection, it is where the file is defined its type, if it is XML/text or other types, moreover, the type of connection is defined if it is FSTP or connecting on a database or other types. The job, it is when we are going to reach to that certain file, as it will be reached daily or hourly, also the job defines if there is a trial mechanism or no.

Due to lack of description of Vodafone's dataset attributes, Telecom Italia dataset is used instead. It is for measuring the interaction level between the users and the mobile phone network. The datasets provide data about the telecommunication activity in Milano city in Italy during month of December, 2013. This dataset has over 90 million records for 10,000 square ids. The original dataset consists of 8 columns which are square id of Milano city that begins from 1 to 10,000 square id, 5 activities which are sms-in activity, sms-out activity, call-in activity, call-out activity and Internet traffic activity, country code, time interval in milliseconds. Modifications were made on this dataset by adding startOfTimeInSeconds, endOfTimeInSeconds, startOfTime, enOfTime, workingHours, and holiday columns to obtain the following: calculating the end of time interval by adding 10 mins, converting start and end of time interval from milliseconds to seconds, obtaining timestamp from time interval in th following format Month-Day-Year H:mm AM/PM. Moreover, specifying holidays including weekends in Italy and the public European holidays in December 2013, by giving value of 1 if it's a holiday and 0 if not. In addition, specifying working hours portion in Italy by giving value 1 to working hours from 8:00AM to 2:00PM, 2 for working hours from 2:00PM to 6:00PM, 3 for working hours from 6:00PM to 12:00AM, and zero for others. Milano city is divided into 10,000 square id GRID as shown in the following figure below.

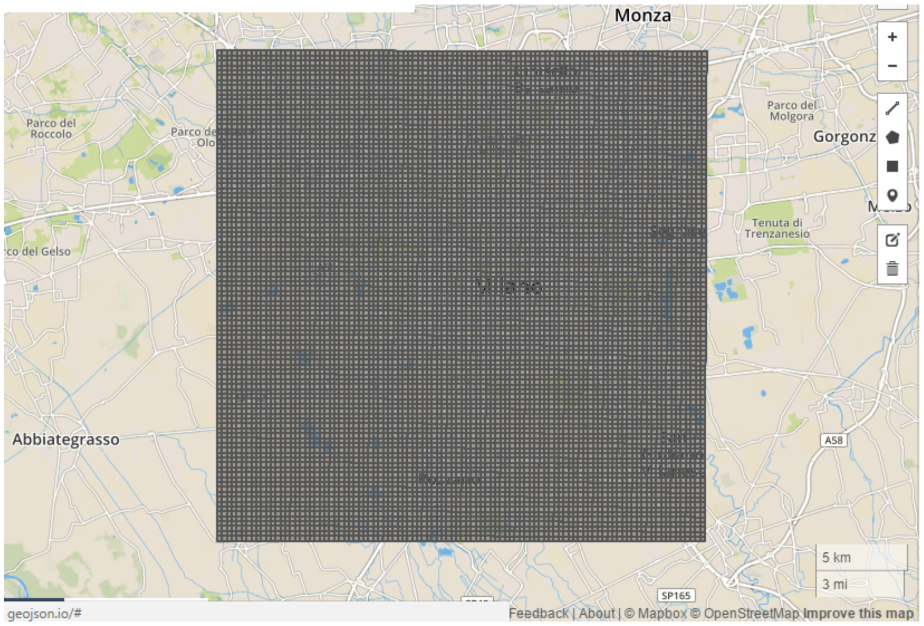


Figure 2: GeoMap for 10,000 Square Id Milano City GRID

A new table is designed having just 75 square id of Milano city GRID, the sum of each activity(sms-in activity, sms-out activity, call-in activity, call-out activity and internet traffic activity), total summation of all activities, and activity level which determine low, moderate and high activity level for each 75 square id.As the 75 square id is divided into three sections: 25 square id for low activity level, 25 square id for moderate activity level and 25 square id for high activity level. This table is arranged ascendingly according to total sum of all activities. The objective of this table is to determine the activity level of each square id .The figure below is showing the 75 square id that is used in the new table.

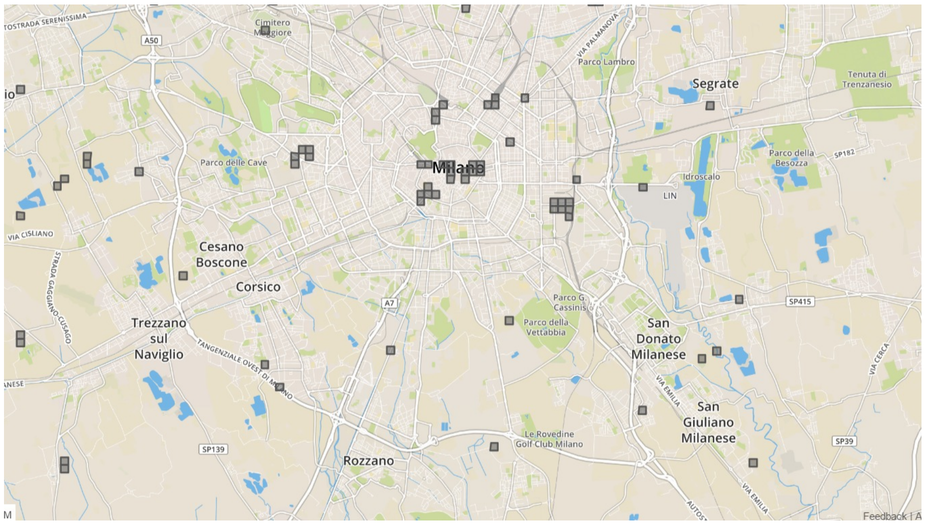


Figure 3: GeoMap for 75 Square Id Milano City GRID

The below figures are graphs illustrating the activity level (lowest/moderate/highest)of all activities for the 75 square ids.

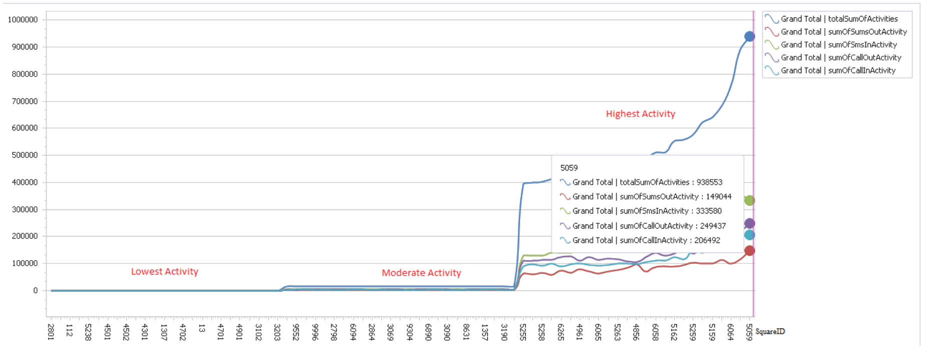


Figure 4: sum Of Activities Per Square ID

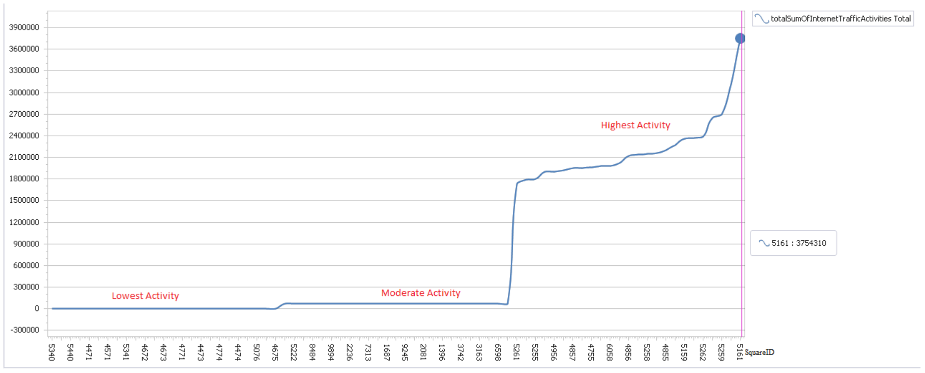


Figure 5: sum Of Internet Traffic Activities Per Square ID

Operations done on Milano dataset as classification and clustering. The Milano dataset is divided into two parts training and testing, 90,000 record for training divided into three section 30,000 records for low activity level, 30,000 records for moderate activity level, 30,000 records for high activity level and 30,000 other records for testing divided into 10,000 for low activity level, 10,000 for moderate activity level, 10,000 for high activity level. The training and testing records are used for classification in support vector machine (SVM) and neural network (NN).

2.1 Classification

For classification, support vector machine (SVM) and neural network (NN) are two strategies that are used to analyze data.

2.1.1 Support Vector Machine

Support Vector Machines (SVM) are supervised learning models with associated learning algorithms that analyze data used for classification. Given binary classes or multi-classes, SVM trains a model forming an optimal hyperplane that separates the classes and then categorizes inputs into one of the classes through the test data. LIBSVM is an integrated software for support vector classification, (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). It supports multi-class classification. C and nu are parameters which help implement a penalty on the misclassifications that are performed while separating the classes. Thus helps in improving the accuracy of the output. C ranges from 0 to infinity and can be a bit hard to estimate and use. A modification to this was the introduction of nu which operates between 0-1 and represents the lower and upper bound on the number of examples that are support vectors and that lie on the wrong side of the hyperplane.

Kernel Functions Equations According to LIBSVM Documentation:

1) Linear: u'*v
2) Polynomial: (gamma*u'*v + coef0)$^{d}egree$
3) Radial basis function: exp(-gamma*—u-v—^{2})
4) sigmoid: tanh(gamma*u'*v + coef0)

-d degree : set degree in kernel function (default 3)

-g gamma : set gamma in kernel function (default 1/num$_{f}eatures$)

-r coef0 : set coef0 in kernel function (default 0)

-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)

-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)

-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)

-m cachesize : set cache memory size in MB (default 100)

-e epsilon : set tolerance of termination criterion (default 0.001)

The k in the -g option means the number of attributes in the input data.

option -v randomly splits the data into n parts and calculates cross validation accuracy/mean squared error on them.

Steps to Use LIBSVM:

1) Open Command Prompt (cmd), browse to libsvm/windows.

2) A certain svm type and kernel function may be specified(default for svm type is C-SVC(multi-class classification) corresponds to parameter value 0, and for kernel function, the default is radial basis function)corresponds to parameter value 2. Further illustration for svm types, and kernel functions are discussed later in this document. Run svm-train command to train a model on certain training data by typing the following command:

```
svm-train.exe "filePath/fileName" or fileName
```

The above command corresponds to the following command:

```
svm-train.exe -s 0 -t 2 "filePath/fileName" or fileName
```

3) A training model is generated. Use this model to test/validate the testing data. By running the following command that includes the testing/validating

data, the generated trained model, and an output file:

svm-predict.exe "filePath/fileName" or fileNameofTestingData modelFile outputFile

After executing the previous commands, an output file is generated, accuracy result.

4) (Optional) In order to visualize the classification process, we run the following command which opens a friendly-graphical user interface application:

svm-toy.exe

5) After the svm-toy application opens up, press the load button to choose a file that contains the data.

6) After data is loaded, type the command in the empty the text box found on the bottom-right of the application screen. The commands are similar to those used in svm-train command; only svm type and kernel function are specified. Command example:

-s 0 -t 1

Dataset is composed of three classes, class(1) contains lowest activities(smsInActivity, smsOut,Activity, callInActivity, callOutActivity, and internetTrafficActivity) for 25 Square IDs, class(2) contains moderate activities for another 25 Square IDs, and class(3) contains highest activities for 25 Square IDs. Hence multi-class classification types are used (C-SVC and nu-SVC)with different kernel functions (Linear, polynomial, radial basis, and sigmoid). For the training dataset, 90 thousand records are used, and for testing 30 thousand records. The below table shows the results for all of the SVM types used and kernel functions used by LIBSVM with their default parameter values and their accuracies for classification.

Kernel-Types	Linear	Polynomial	Radial Basis	Sigmoid
SVM-Types				
C-SVC (multi-class classification)	48.3035%	29.091%	65.8629%	35.957%
nu-SVC (multi-class classification)	46.1328%	40.7294%	65.3186%	43.2741%

According to the accuracy results, C-SVC type with Radial Basis kernel function achieved the highest accuracy. After further enhancement and manipulation for gamma parameter of Radial Basis kernel function, the following accuracies are achieved.

C-SVC, Kernel-Types	Radial Basis Kernel Function
gamma= 0.2(default)	65.8269%
gamma= 0.3	66.3372%
gamma= 0.6	69.0255%
gamma= 0.98	70.4114%
gamma= 0.99	70.4415%
gamma= 1.3	70.5884%
gamma= 1.8	71.1027%

The following figure shows the dataset before classification.

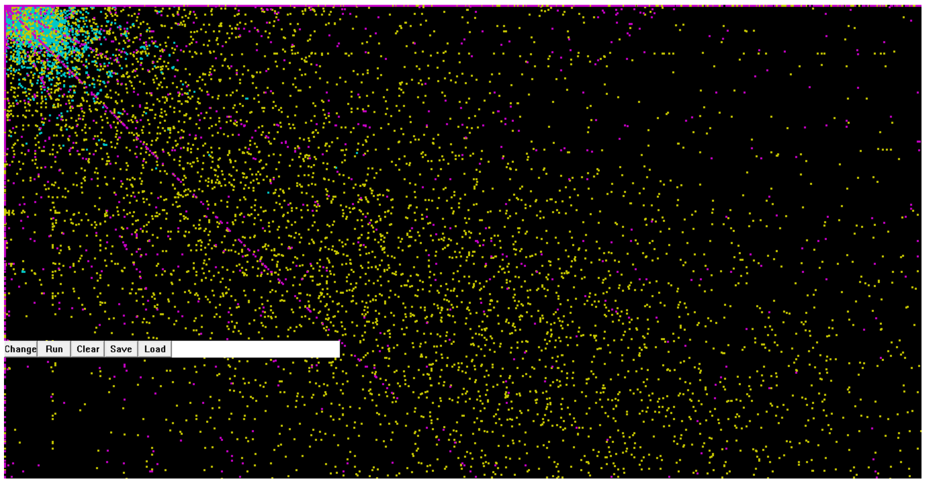
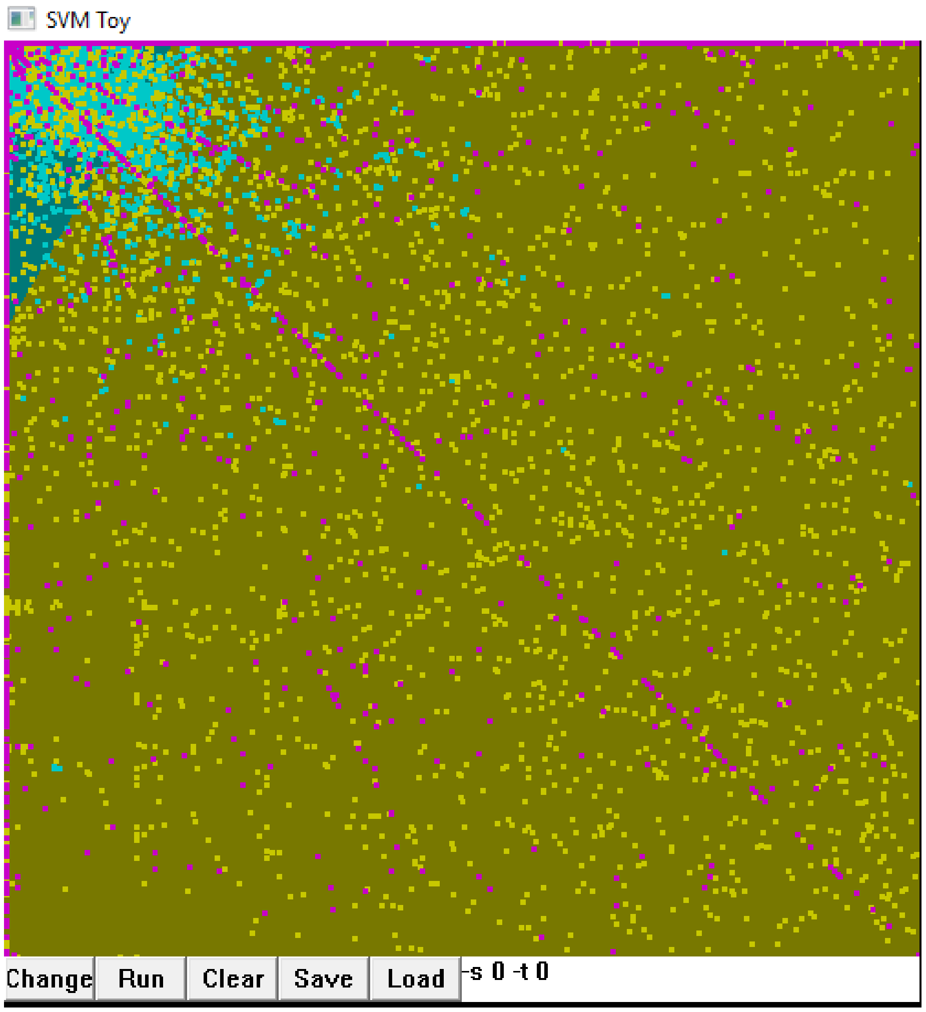


Figure 6: Training data before classification

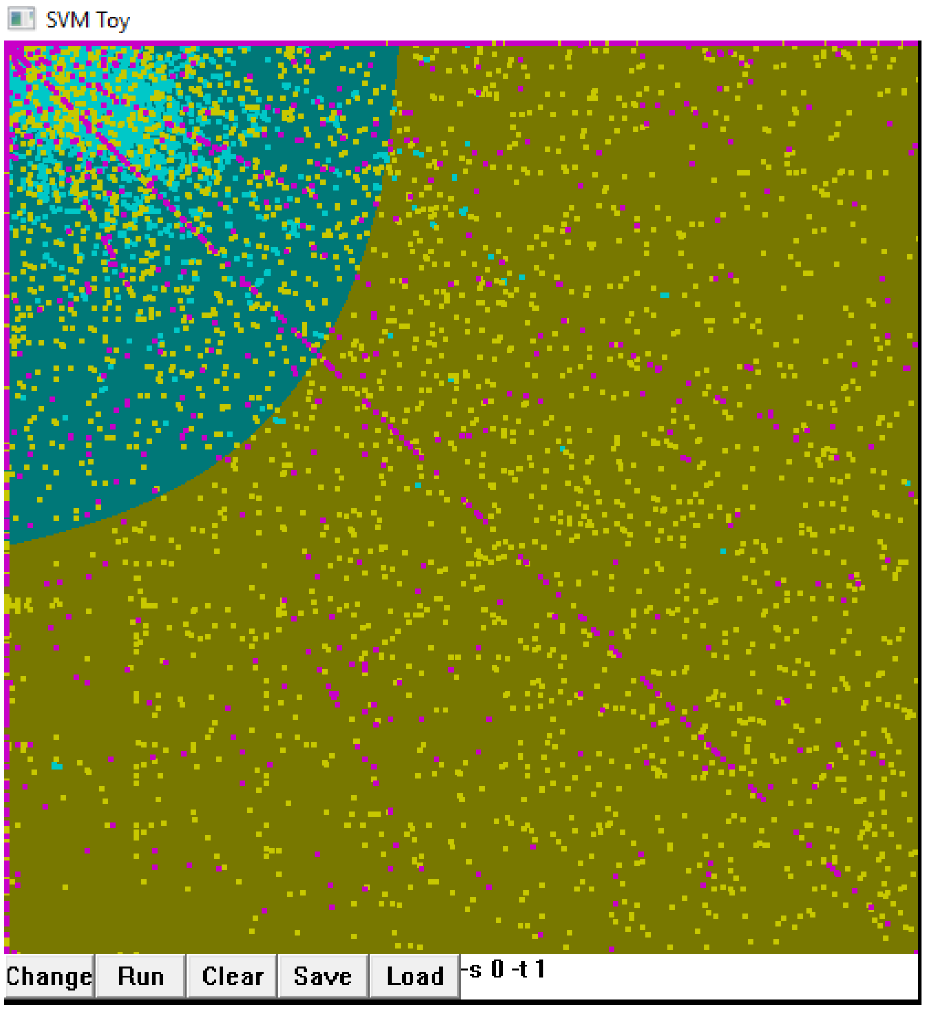
The following figure shows dataset after classification(C-SVC type, and Linear Kernel function). The linear kernel function produces a straight hyperplane that failed to separate the dots into three different categories.



```
-s 0 -t 0
```

Figure 7: Training data after classification with C-SVC type, and Linear Kernel function

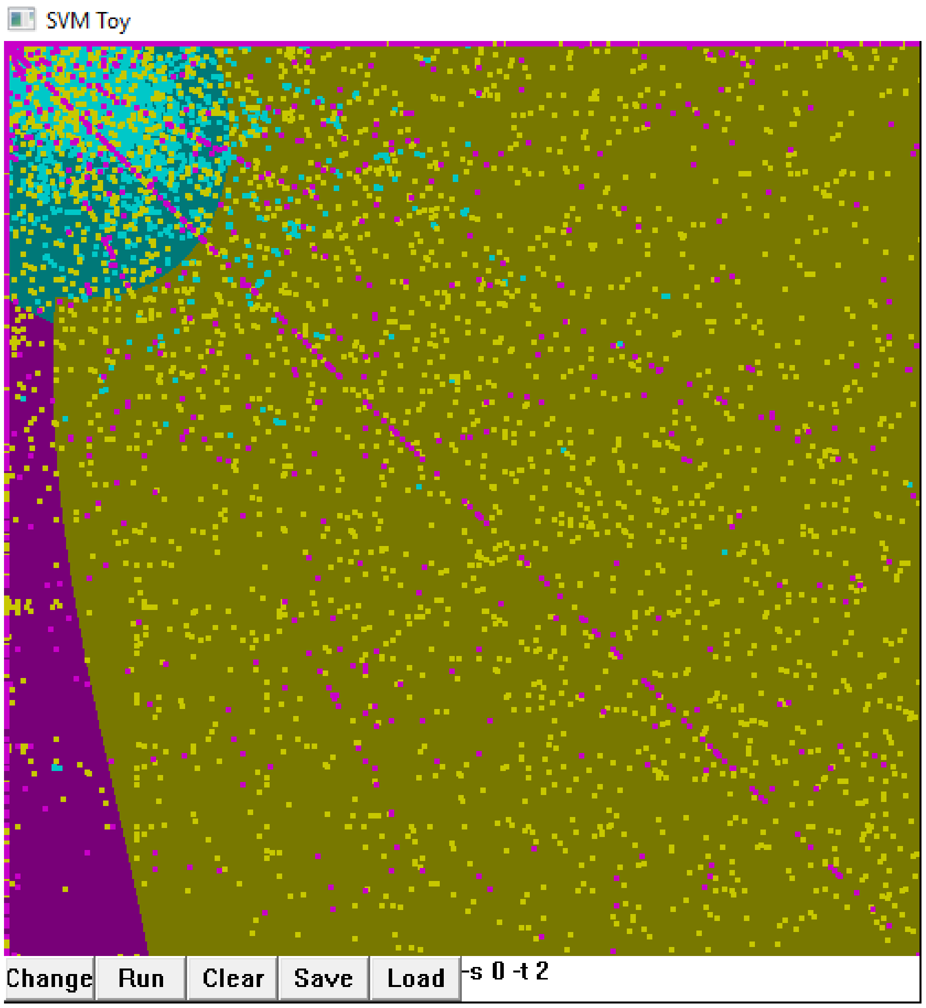
The following figure shows the dataset after classification(C-SVC, and polynomial kernel function with default parameter values). The polynomial function produced a hyperplane with wide margins causing an issue hence resulting in low accuracy; dots are misclassified. e.g: some of the yellow dots are classified as blue dots.



-s 0 -t 1

Figure 8: Training data after classification with C-SVC type, and polynomial kernel function with default parameter values

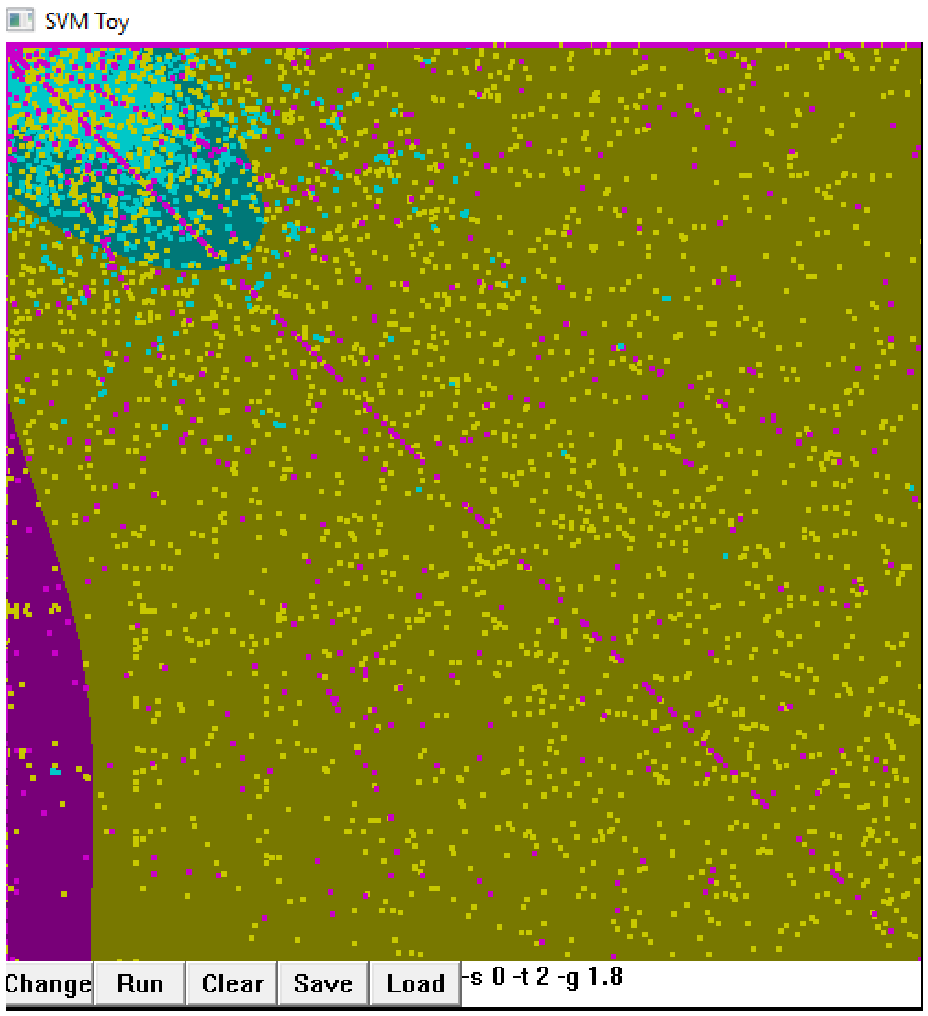
The following figure shows the dataset after classification (C-SVC type, and Radial Basis Kernel function with default gamma value). Radial basis kernel function produced hyperplanes that almost correctly separated all dots into three different categories with a few missclassified dots.



-s 0 -t 2

Figure 9: Training data after classification with C-SVC type, and Radial Basis Kernel function with default gamma value

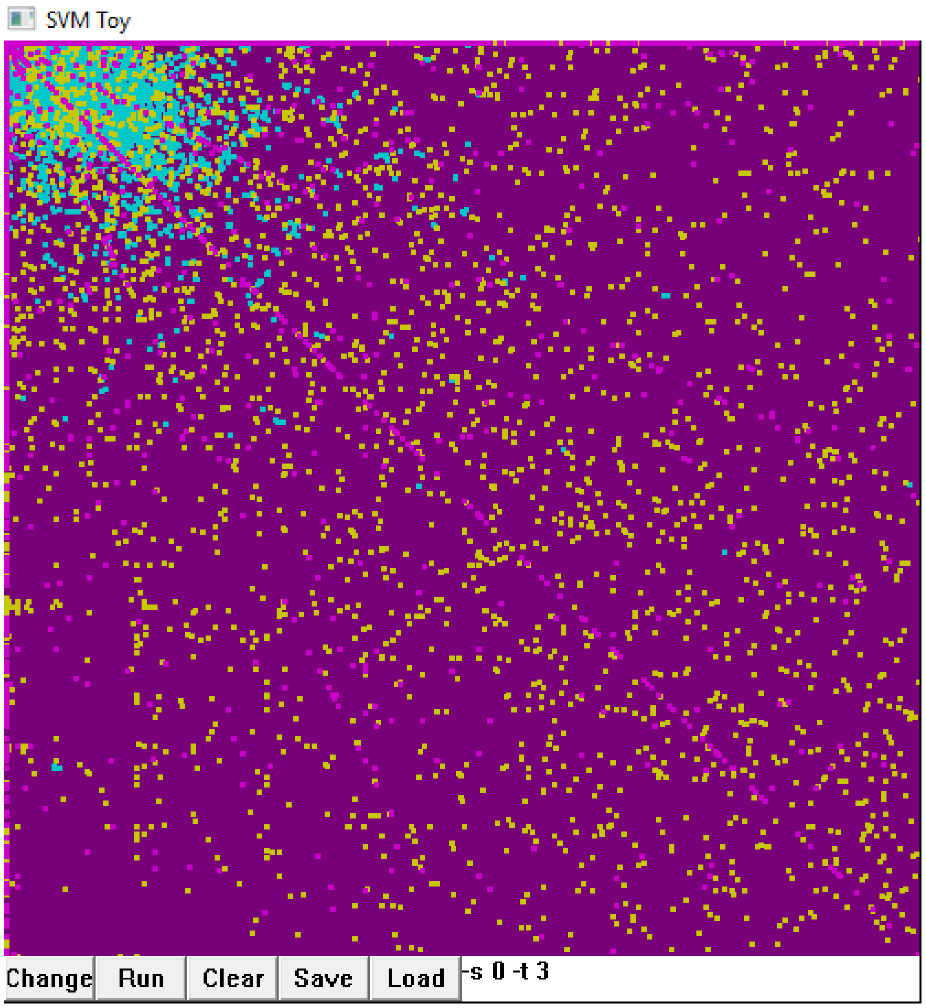
The following figure shows the dataset after classification and after manipulating gamma parameter value(C-SVC type, and Radial Basis Kernel function with 1.8 gamma value). The enhancement done by changing the gamma value resulted in higher accuracies. The highest one was with gamma = 1.8. The hyperplanes produced, separated the three categories better than with the default gamma parameter, because of a smaller margin, number of missclassified dots decreased hence higher accuracy.



-s 0 -t 2 -g 1.8

Figure 10: Training data after classification with C-SVC type, and Radial Basis Kernel function with 1.8 gamma value

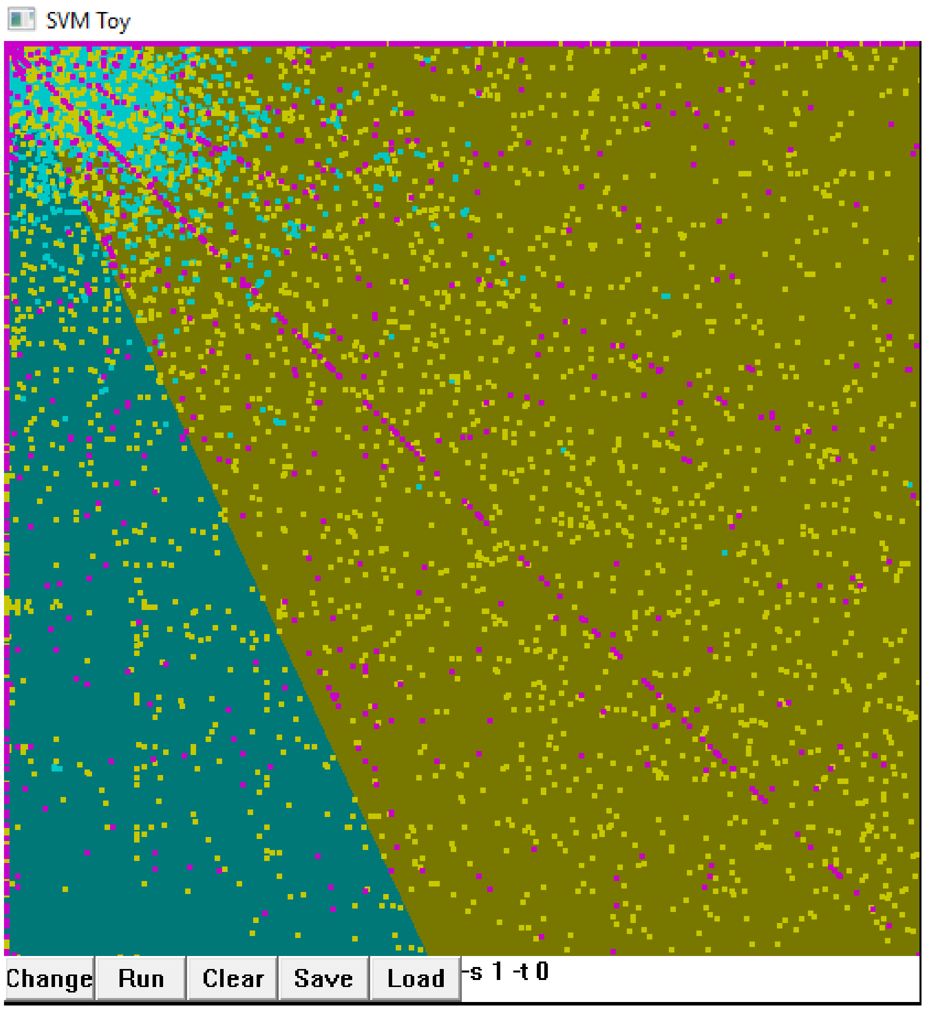
The following figure shows the dataset after classification(C-SVC type, and sigmoid kernel function with default parameter values).



-s 0 -t 3

Figure 11: Training data after classification with C-SVC type, and sigmoid kernel function with default parameter values

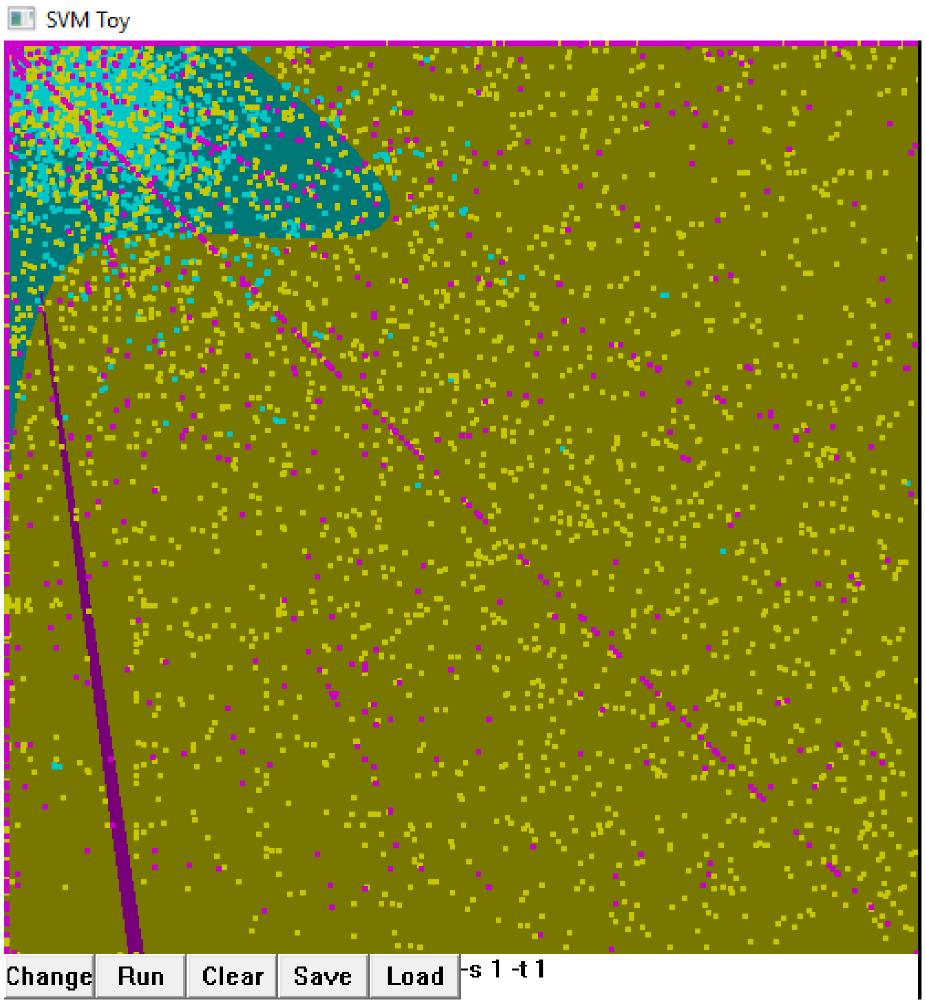
The following figure shows the dataset after classification(nu-SVC type, and Linear kernel function). The linear kernel function produces a straight hyperplane that failed to separate the dots into three different categories.



-s 1 -t 0

Figure 12: Training data after classification(nu-SVC type, and Linear kernel function)

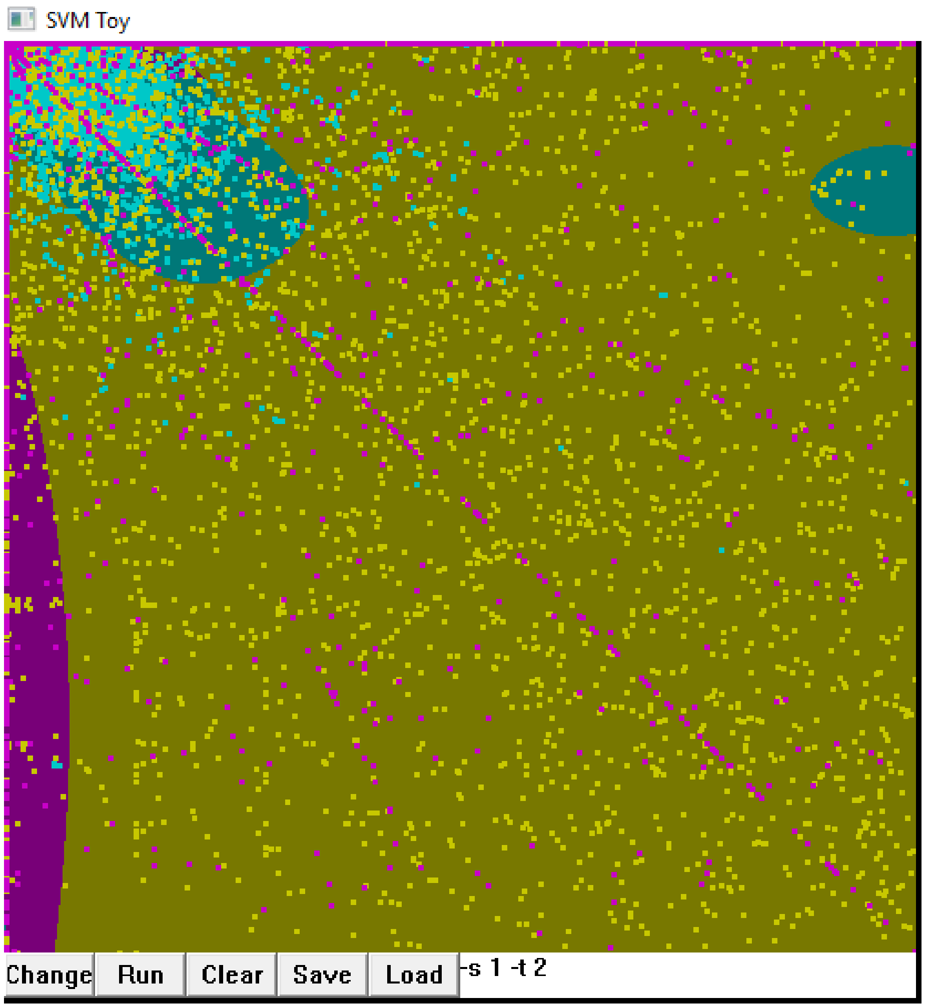
The following figure shows the dataset after classification(nu-SVC type, and polynomial kernel function with default parameter values). The polynomial function produced a hyperplane with wide margins causing an issue hence resulting in low accuracy; dots are misclassified. e.g: some of the yellow dots are classified as blue dots, some of the purple dots are classified as blue dots, some the purple dots as yellow ones.



-s 1 -t 1

Figure 13: Training data after classification(nu-SVC type, and polynomial kernel function with default parameter values)

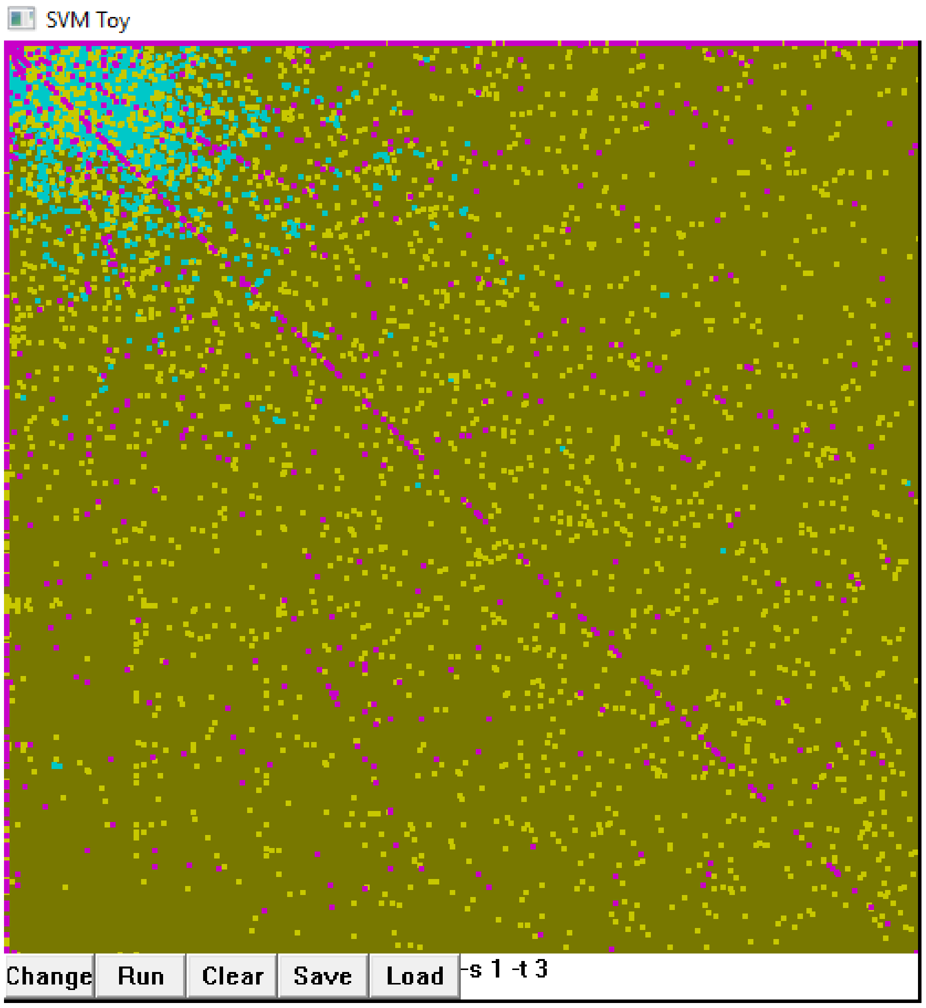
The following figure shows the dataset after classification(nu-SVC type, and radial basis kernel function with default parameter values). Radial basis kernel function produced hyperplanes that almost correctly separated all dots into three different categories with a few missclassified dots.



-s 1 -t 2

Figure 14: Training data after classification(nu-SVC type, and radial basis kernel function with default parameter value)

The following figure shows the dataset after classification(nu-SVC type, and sigmoid kernel function with default parameter values).



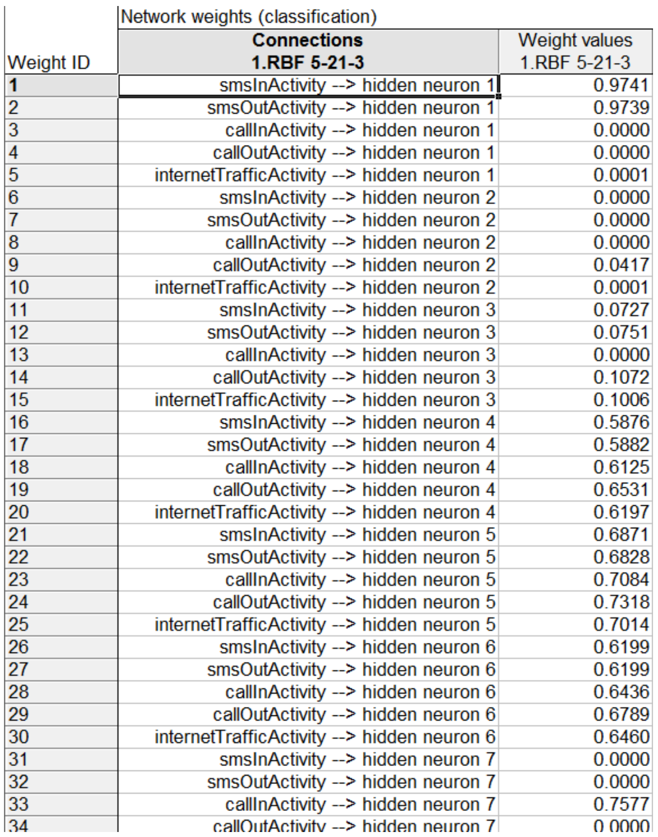
-s 1 -t 3

Figure 15: Training data after classification(nu-SVC type, and sigmoid kernel function with default parameter value)

2.1.2 Neural Network

A neural network is a supervised learning models. It consists of units (neurons), arranged in layers, which convert an input vector into some output. Each unit takes an input, applies a (often nonlinear) function to it and then passes the output on to the next layer. Weightings are applied to the signals passing from one unit to another, and it is these weightings which are tuned in the training phase to adapt a neural network to the particular problem at hand. The used algorithm in neural network classification is Radial Basis Function (RBF). The RBF consists of 3 layers; Input layer, Hidden layer, Output layer. Each unit in hidden layer consists of a radial basis function centered on a point with the same dimensions as predictor variables. The output layer has weighted sum of outputs from the hidden layer to form the network output. RBF is similar to k-nearest neighbor models. The basic idea is that a predicted target value of an item is likely to be about the same as other items that have a close value of predictor variables. We use euclidean distance to calculate a distance from the point being evaluated to the center of each unit and RBF is applied to the distance to compute the weight for each unit. Here, the resulted accuracy is made by selecting the activity level column with the other columns. These are the weights and the hidden layers used for classification:

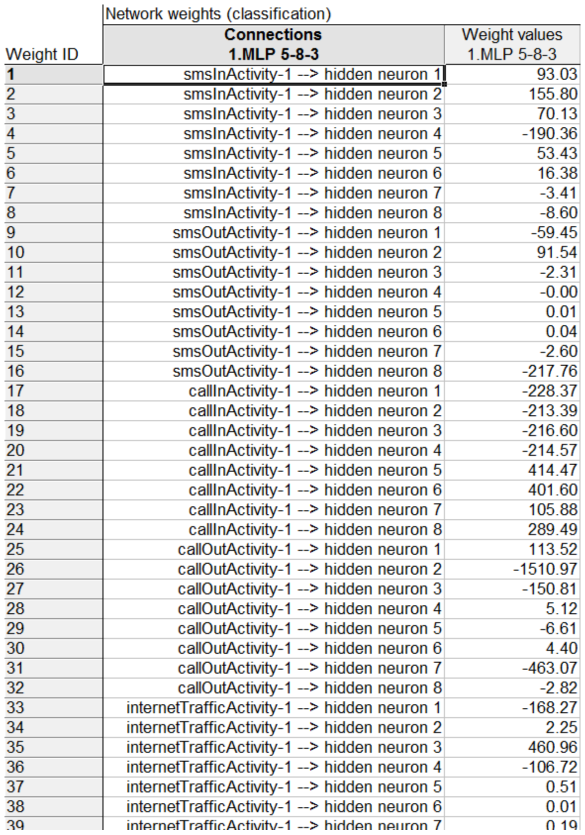
Type	Selected Algorithm	Classifier Accuracy Percentage
Train	RBF	71.2
Test	RBF	72.35

(1).png 

Weight ID	Network weights (classification) Connections 1.RBF 5-21-3	Weight values 1.RBF 5-21-3
1	smsInActivity --> hidden neuron 1	0.9741
2	smsOutActivity --> hidden neuron 1	0.9739
3	callInActivity --> hidden neuron 1	0.0000
4	callOutActivity --> hidden neuron 1	0.0000
5	internetTrafficActivity --> hidden neuron 1	0.0001
6	smsInActivity --> hidden neuron 2	0.0000
7	smsOutActivity --> hidden neuron 2	0.0000
8	callInActivity --> hidden neuron 2	0.0000
9	callOutActivity --> hidden neuron 2	0.0417
10	internetTrafficActivity --> hidden neuron 2	0.0001
11	smsInActivity --> hidden neuron 3	0.0727
12	smsOutActivity --> hidden neuron 3	0.0751
13	callInActivity --> hidden neuron 3	0.0000
14	callOutActivity --> hidden neuron 3	0.1072
15	internetTrafficActivity --> hidden neuron 3	0.1006
16	smsInActivity --> hidden neuron 4	0.5876
17	smsOutActivity --> hidden neuron 4	0.5882
18	callInActivity --> hidden neuron 4	0.6125
19	callOutActivity --> hidden neuron 4	0.6531
20	internetTrafficActivity --> hidden neuron 4	0.6197
21	smsInActivity --> hidden neuron 5	0.6871
22	smsOutActivity --> hidden neuron 5	0.6828
23	callInActivity --> hidden neuron 5	0.7084
24	callOutActivity --> hidden neuron 5	0.7318
25	internetTrafficActivity --> hidden neuron 5	0.7014
26	smsInActivity --> hidden neuron 6	0.6199
27	smsOutActivity --> hidden neuron 6	0.6199
28	callInActivity --> hidden neuron 6	0.6436
29	callOutActivity --> hidden neuron 6	0.6789
30	internetTrafficActivity --> hidden neuron 6	0.6460
31	smsInActivity --> hidden neuron 7	0.0000
32	smsOutActivity --> hidden neuron 7	0.0000
33	callInActivity --> hidden neuron 7	0.7577
34	callOutActivity --> hidden neuron 7	0.0000

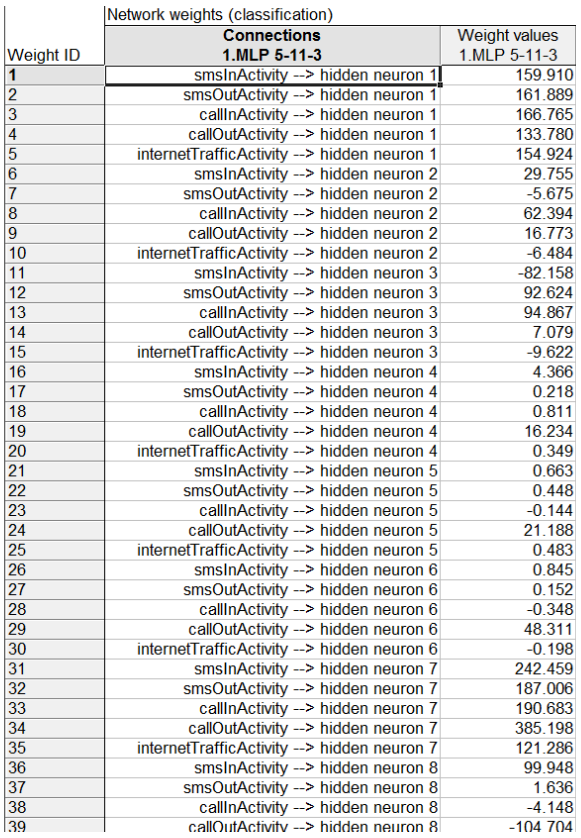
Figure 16: Weights and hidden layers for RBF

(2).png



Weight ID	Network weights (classification) Connections 1.MLP 5-8-3	Weight values 1.MLP 5-8-3
1	smsInActivity-1 --> hidden neuron 1	93.03
2	smsInActivity-1 --> hidden neuron 2	155.80
3	smsInActivity-1 --> hidden neuron 3	70.13
4	smsInActivity-1 --> hidden neuron 4	-190.36
5	smsInActivity-1 --> hidden neuron 5	53.43
6	smsInActivity-1 --> hidden neuron 6	16.38
7	smsInActivity-1 --> hidden neuron 7	-3.41
8	smsInActivity-1 --> hidden neuron 8	-8.60
9	smsOutActivity-1 --> hidden neuron 1	-59.45
10	smsOutActivity-1 --> hidden neuron 2	91.54
11	smsOutActivity-1 --> hidden neuron 3	-2.31
12	smsOutActivity-1 --> hidden neuron 4	-0.00
13	smsOutActivity-1 --> hidden neuron 5	0.01
14	smsOutActivity-1 --> hidden neuron 6	0.04
15	smsOutActivity-1 --> hidden neuron 7	-2.60
16	smsOutActivity-1 --> hidden neuron 8	-217.76
17	callInActivity-1 --> hidden neuron 1	-228.37
18	callInActivity-1 --> hidden neuron 2	-213.39
19	callInActivity-1 --> hidden neuron 3	-216.60
20	callInActivity-1 --> hidden neuron 4	-214.57
21	callInActivity-1 --> hidden neuron 5	414.47
22	callInActivity-1 --> hidden neuron 6	401.60
23	callInActivity-1 --> hidden neuron 7	105.88
24	callInActivity-1 --> hidden neuron 8	289.49
25	callOutActivity-1 --> hidden neuron 1	113.52
26	callOutActivity-1 --> hidden neuron 2	-1510.97
27	callOutActivity-1 --> hidden neuron 3	-150.81
28	callOutActivity-1 --> hidden neuron 4	5.12
29	callOutActivity-1 --> hidden neuron 5	-6.61
30	callOutActivity-1 --> hidden neuron 6	4.40
31	callOutActivity-1 --> hidden neuron 7	-463.07
32	callOutActivity-1 --> hidden neuron 8	-2.82
33	internetTrafficActivity-1 --> hidden neuron 1	-168.27
34	internetTrafficActivity-1 --> hidden neuron 2	2.25
35	internetTrafficActivity-1 --> hidden neuron 3	460.96
36	internetTrafficActivity-1 --> hidden neuron 4	-106.72
37	internetTrafficActivity-1 --> hidden neuron 5	0.51
38	internetTrafficActivity-1 --> hidden neuron 6	0.01
39	internetTrafficActivity-1 --> hidden neuron 7	0.19

Figure 17: Weights and hidden layers for MLP



Network weights (classification)

Weight ID	Connections 1.MLP 5-11-3	Weight values 1.MLP 5-11-3
1	smsInActivity --> hidden neuron 1	159.910
2	smsOutActivity --> hidden neuron 1	161.889
3	callInActivity --> hidden neuron 1	166.765
4	callOutActivity --> hidden neuron 1	133.780
5	internetTrafficActivity --> hidden neuron 1	154.924
6	smsInActivity --> hidden neuron 2	29.755
7	smsOutActivity --> hidden neuron 2	-5.675
8	callInActivity --> hidden neuron 2	62.394
9	callOutActivity --> hidden neuron 2	16.773
10	internetTrafficActivity --> hidden neuron 2	-6.484
11	smsInActivity --> hidden neuron 3	-82.158
12	smsOutActivity --> hidden neuron 3	92.624
13	callInActivity --> hidden neuron 3	94.867
14	callOutActivity --> hidden neuron 3	7.079
15	internetTrafficActivity --> hidden neuron 3	-9.622
16	smsInActivity --> hidden neuron 4	4.366
17	smsOutActivity --> hidden neuron 4	0.218
18	callInActivity --> hidden neuron 4	0.811
19	callOutActivity --> hidden neuron 4	16.234
20	internetTrafficActivity --> hidden neuron 4	0.349
21	smsInActivity --> hidden neuron 5	0.663
22	smsOutActivity --> hidden neuron 5	0.448
23	callInActivity --> hidden neuron 5	-0.144
24	callOutActivity --> hidden neuron 5	21.188
25	internetTrafficActivity --> hidden neuron 5	0.483
26	smsInActivity --> hidden neuron 6	0.845
27	smsOutActivity --> hidden neuron 6	0.152
28	callInActivity --> hidden neuron 6	-0.348
29	callOutActivity --> hidden neuron 6	48.311
30	internetTrafficActivity --> hidden neuron 6	-0.198
31	smsInActivity --> hidden neuron 7	242.459
32	smsOutActivity --> hidden neuron 7	187.006
33	callInActivity --> hidden neuron 7	190.683
34	callOutActivity --> hidden neuron 7	385.198
35	internetTrafficActivity --> hidden neuron 7	121.286
36	smsInActivity --> hidden neuron 8	99.948
37	smsOutActivity --> hidden neuron 8	1.636
38	callInActivity --> hidden neuron 8	-4.148
39	callOutActivity --> hidden neuron 8	-104.704

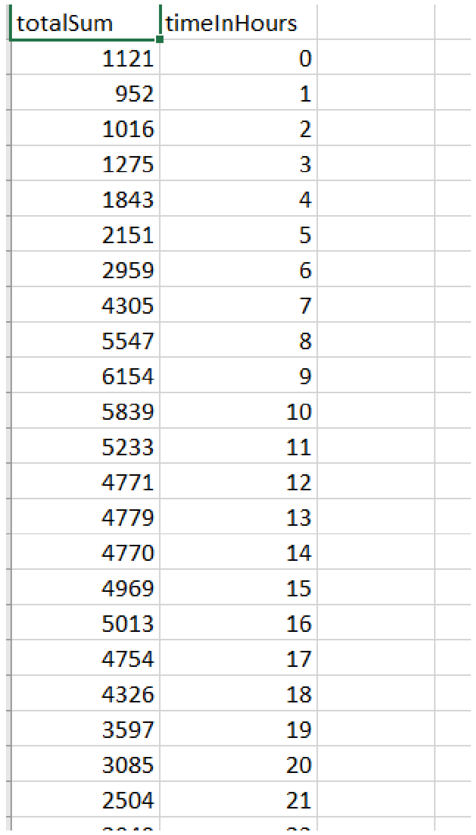
(3).png

Figure 18: Weights and hidden layers for MLP

2.2 Clustering

Using tool named KNIME to cluster the data by K-means.k-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters). Input data are: I m examples x 1 , .., x m , and I K , the number of desired clusters I Clusters represented by cluster centers 1, .., K I Given centers 1, .., K , each center defines a cluster: the subset of inputs x i that are closer to it than to other centers

Dataset consists of two columns (totalsum , timeInHours). totalsum stands for all activities (smsInActivity, smsOut,Activity, callInActivity, callOutActivity, internetTrafficActivity), timeInHours stands for start time intervals, according to these columns, k-means clustering is used to know when the peak times happens through activities.



totalSum	timeInHours
1121	0
952	1
1016	2
1275	3
1843	4
2151	5
2959	6
4305	7
5547	8
6154	9
5839	10
5233	11
4771	12
4779	13
4770	14
4969	15
5013	16
4754	17
4326	18
3597	19
3085	20
2504	21

Figure 19: Sample from data

This figure shows how to perform a clustering of the dataset using the k-Means node.

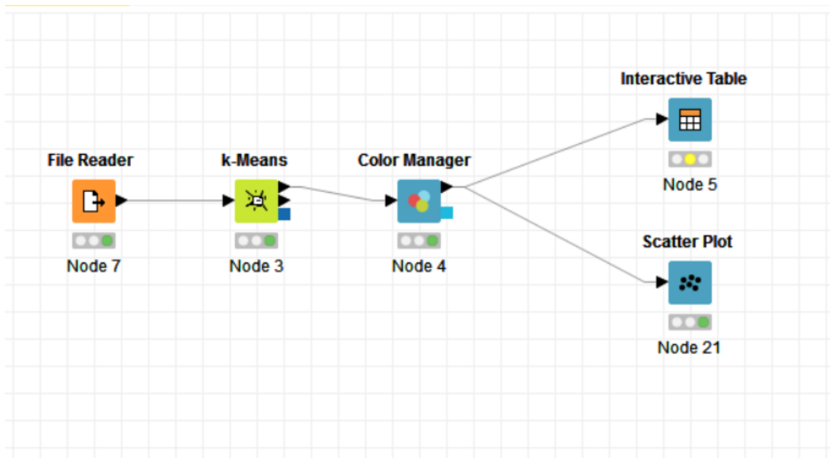
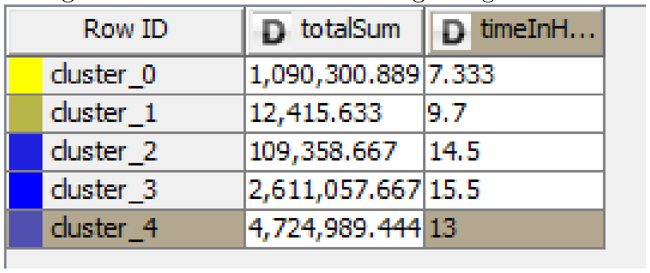


Figure 20: Workflow for clustring using K-means.



Row ID	D totalSum	D timeInH...
🟨 cluster_0	1,090,300.889	7.333
🟫 cluster_1	12,415.633	9.7
🟦 cluster_2	109,358.667	14.5
🟦 cluster_3	2,611,057.667	15.5
🟪 cluster_4	4,724,989.444	13

Figure 21: Interactive Table

3 System Architecture

3.1 Architectural Design

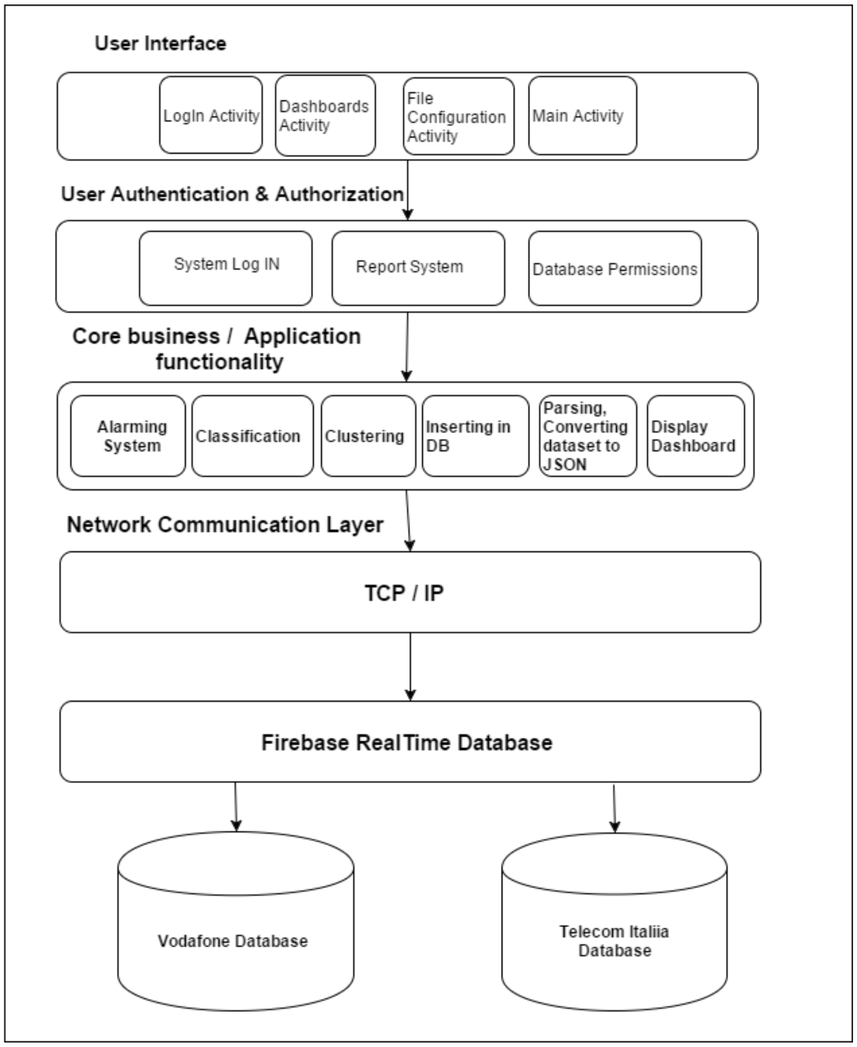


Figure 22: Layered Architecture Diagram

Layered architecture approach is used. The layered approach is composed of five layers, User Interface, User Authorization and Authentication, Core Business/Application functionality, Network Communication and database layers .

3.1.1 User Interface Layer

User Interface layer is the first layer. There are two graphical user interfaces, one for a website, and the other is for a mobile application.

3.1.2 User Authentication and Authorization Layer

User Authentication and Authorization layer is the second layer after User Interface layer. It's composed of MainActivity, LoginActivity, DashboardActivity, and ConfigurationActivity.

3.1.3 Core Business/ Application Functionality Layer

Core Business/ Application layer Functionality is the third layer. It is composed of the technical part of the system, it shows the functionalities provided by the system to the user as Alarming system, classification, parsing and converting to json, inserting into Firebase database, and dashboards.

3.1.4 Network Communication Layer

The fourth layer is the network communication layer, as TCP/IP will be used

3.1.5 Data Layer

Data layer is the fifth layer. It contains the databases that are used by the system.

3.2 Decomposition Description

3.2.1 Activity Diagram

The activity diagram describes the life cycle of the application as shown in the figure below. Our proposed system is a mobile application that takes a file, parse and save it in the DB.Alert the user by mail with the column values exceeded certain threshold and allow the user to graph charts.

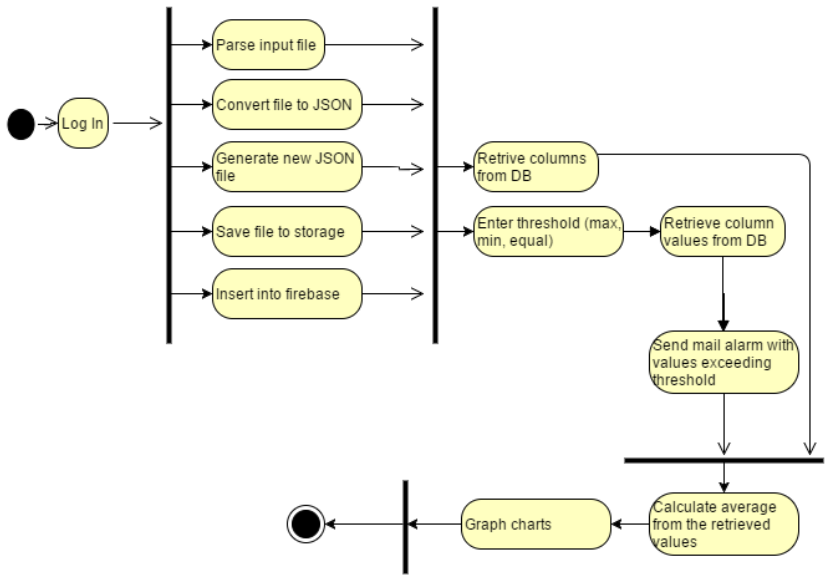

Log In
Parse input file
Convert file to JSON
Generate new JSON file
Save file to storage
Insert into firebase
Retrive columns from DB
Enter threshold (max, min, equal)
Retrieve column values from DB
Send mail alarm with values exceeding threshold
Calculate average from the retrieved values
Graph charts

Figure 23: Activity Diagram

3.2.2 Sequence Diagram

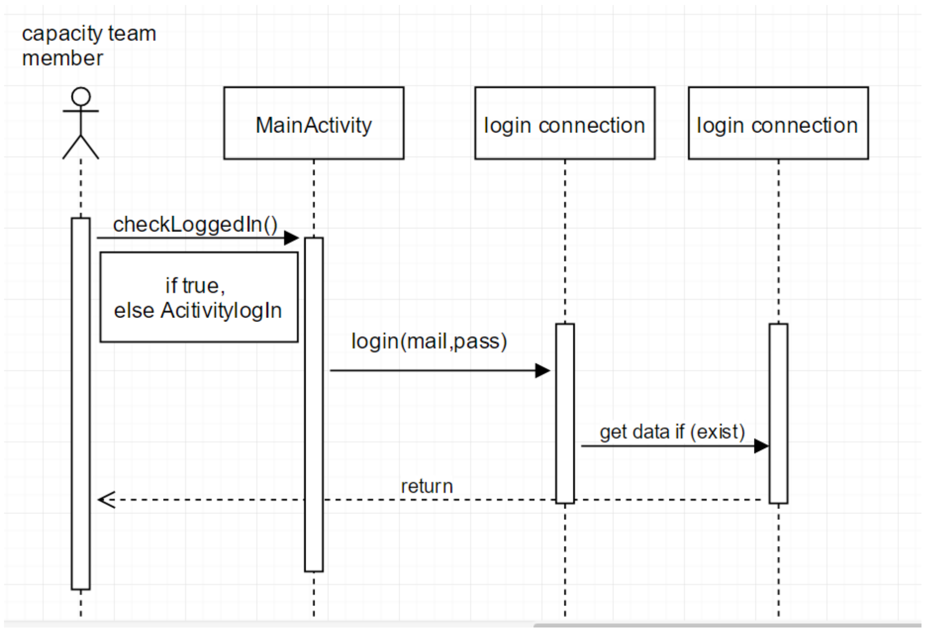


Figure 24: login sequence diagram
Rationale: the user should login if her has an account, if not then go to activity login page.
Input: email, password.
output:next activity page.

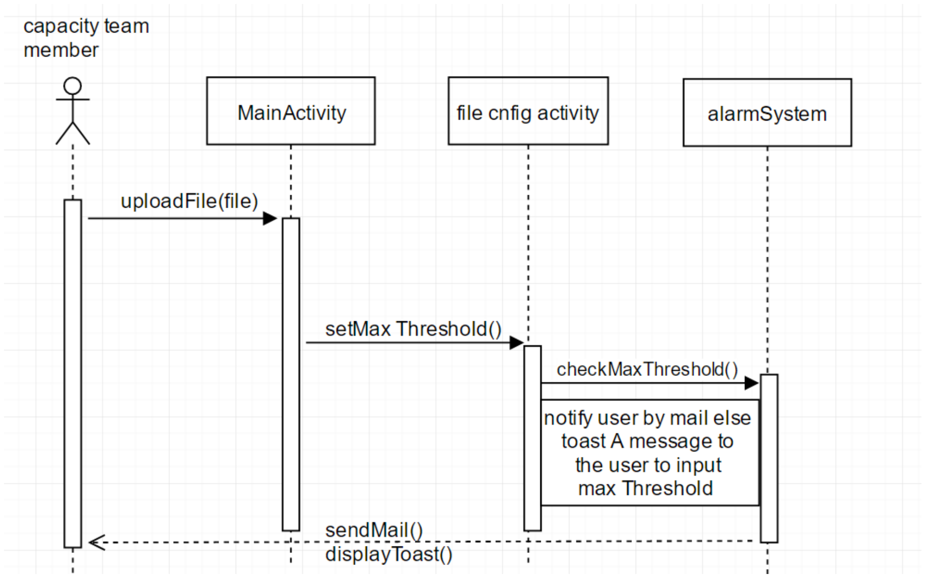


Figure 25: threshold activity sequence diagram
Rationale: the user should upload the file from database(firebase) then set the max threshold and send to the user an alert mail. if there's no max threshold then a toast message is previewed.
Input: max threshold.
output: a notifying mail.

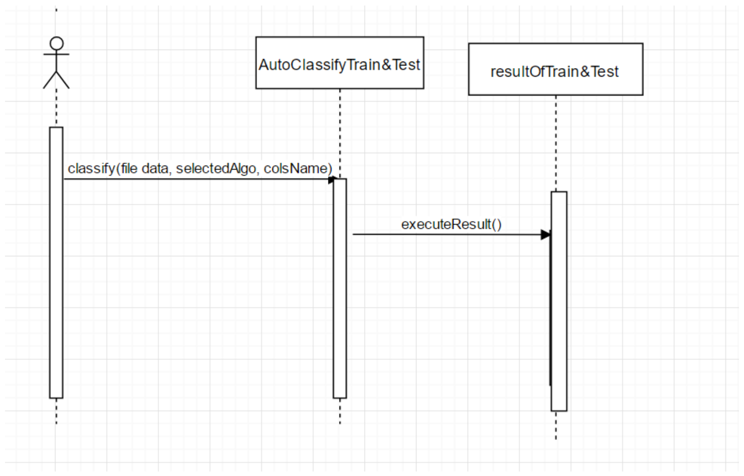


Figure 26: Classification sequence diagram using Neural Network
Rationale: Training, Testing and modelling dataset.
Input: Dataset.
output: Classified data.

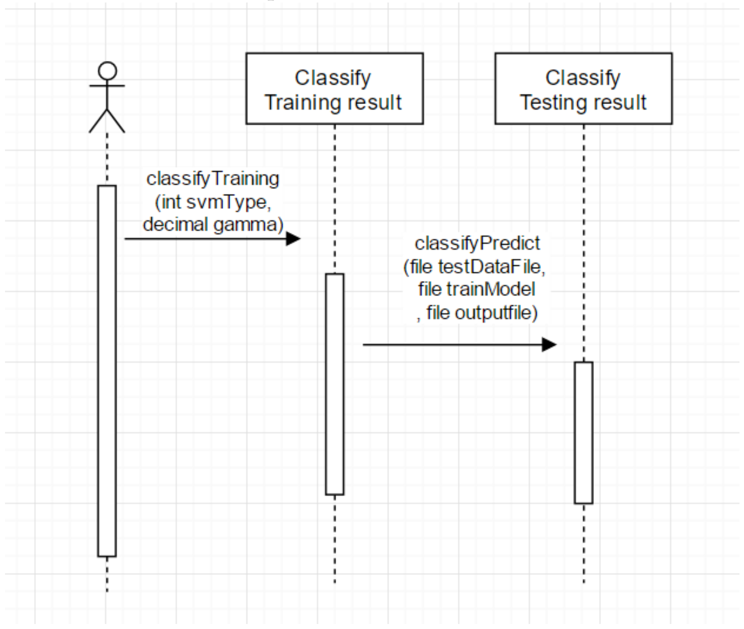

Classify Training result
Classify Testing result
classifyTraining (int svmType, decimal gamma)
classifyPredict (file testDataFile, file trainModel , file outputfile)

Figure 27: Classification sequence Diagram using SVM
Rationale: Training, Testing and modelling dataset.
Input: Dataset.
output: Classified data.

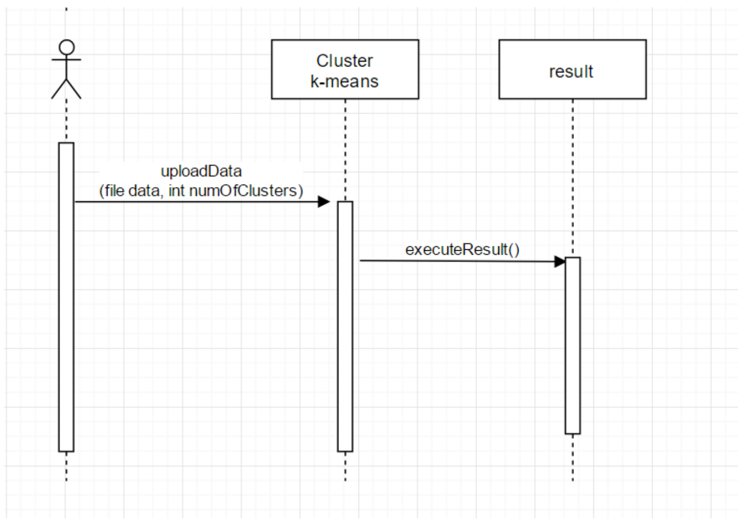


Figure 28: Clustering sequence Diagram
Rationale: uploading file.
Input: Dataset.
output: Clustered data.

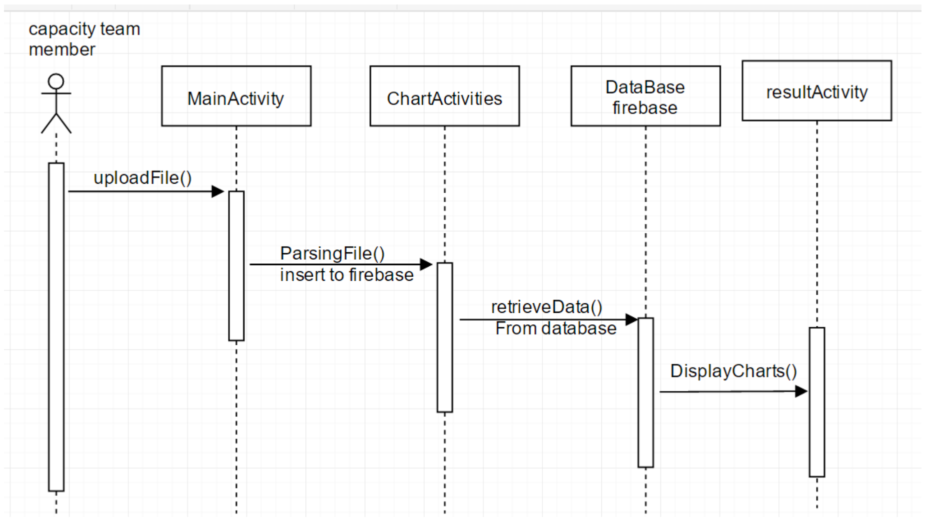


Figure 29: MainActivity sequence Diagram
Rationale: the user should upload the file from database(firebase) then the file is parsed automatically to view the data in order to display charts.
Input: uploading the file.
output: diplay charts.

3.3 Design Rationale

Capacity Monitoring Tool architecture diagram used the layered architecture design as each stage is depending on the stage before it.

4 Data Design

4.1 Data Description

Telecom Italia and Vodafone datasets are used, the data comes in comma-delimited format which are converted to JSON format in order to store the data in Firebase Realtime Database. The hierarchy of the JSON stored in Firebase database is as shown in the following figures:

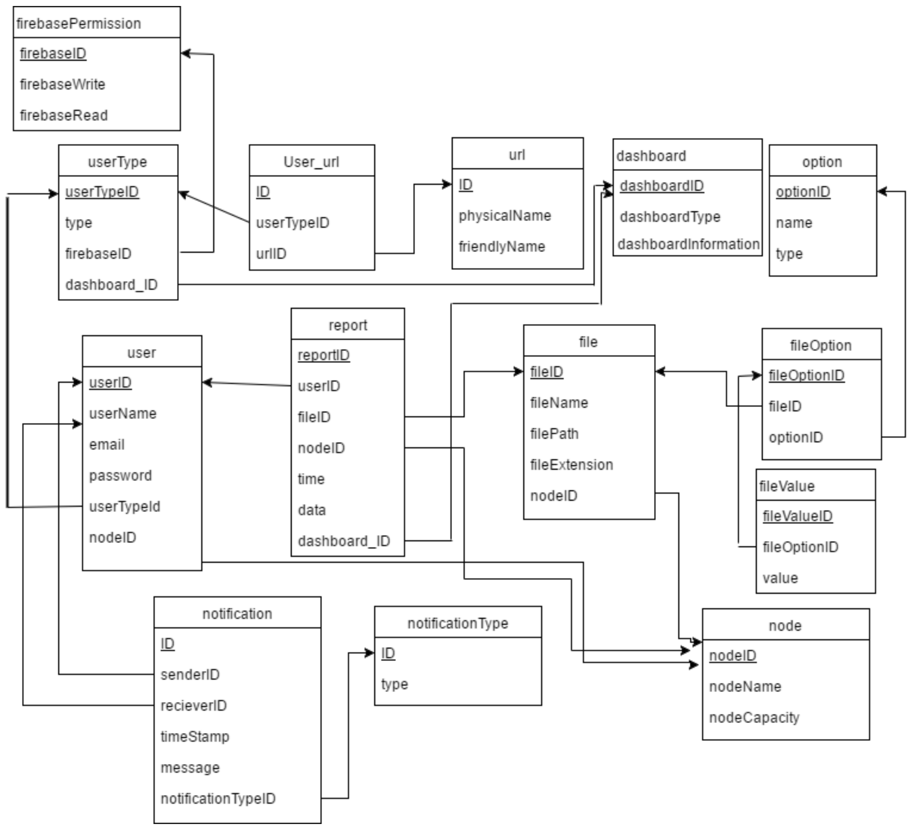


Figure 30: Database

4.1.1 Vodafone Dataset Firebase Realtime Database

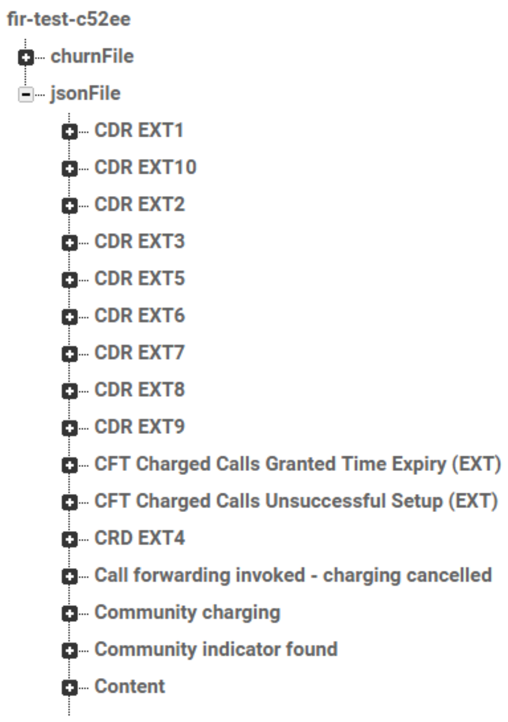
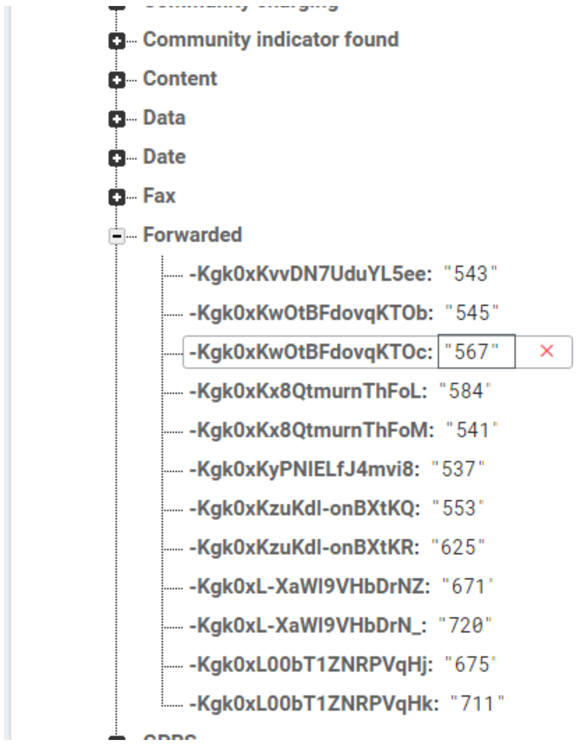

fir-test-c52ee
churnFile
jsonFile
CDR EXT1
CDR EXT10
CDR EXT2
CDR EXT3
CDR EXT5
CDR EXT6
CDR EXT7
CDR EXT8
CDR EXT9
CFT Charged Calls Granted Time Expiry (EXT)
CFT Charged Calls Unsuccessful Setup (EXT)
CRD EXT4
Call forwarding invoked - charging cancelled
Community charging
Community indicator found
Content

Figure 31: Structure/hierarchy of Vodafone Dataset Firebase Realtime Database

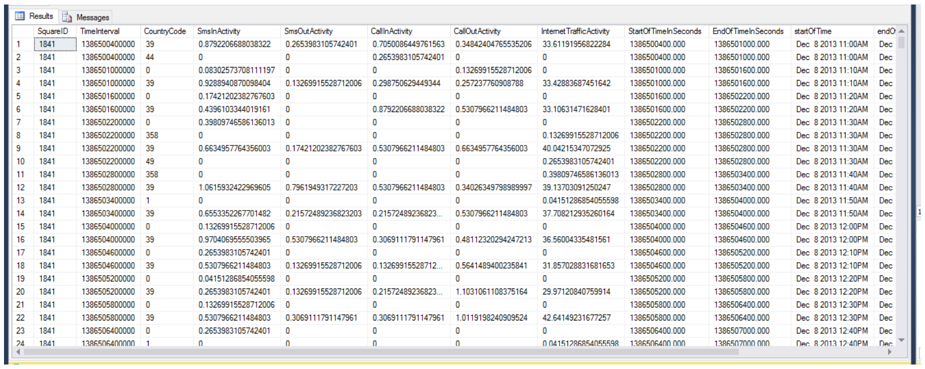


```
+ Community indicator found
+ Content
+ Data
+ Date
+ Fax
- Forwarded
    -Kgk0xKvvDN7UduYL5ee: "543"
    -Kgk0xKwOtBFdovqKTOb: "545"
    -Kgk0xKwOtBFdovqKTOc: "567"  ×
    -Kgk0xKx8QtmurnThFoL: "584"
    -Kgk0xKx8QtmurnThFoM: "541"
    -Kgk0xKyPNIELfJ4mvi8: "537"
    -Kgk0xKzuKdl-onBXtKQ: "553"
    -Kgk0xKzuKdl-onBXtKR: "625"
    -Kgk0xL-XaWI9VHbDrNZ: "671"
    -Kgk0xL-XaWI9VHbDrN_: "720"
    -Kgk0xL00bT1ZNRPVqHj: "675"
    -Kgk0xL00bT1ZNRPVqHk: "711"
```

Figure 32: Structure/hierarchy of Vodafone Dataset Firebase Realtime Database

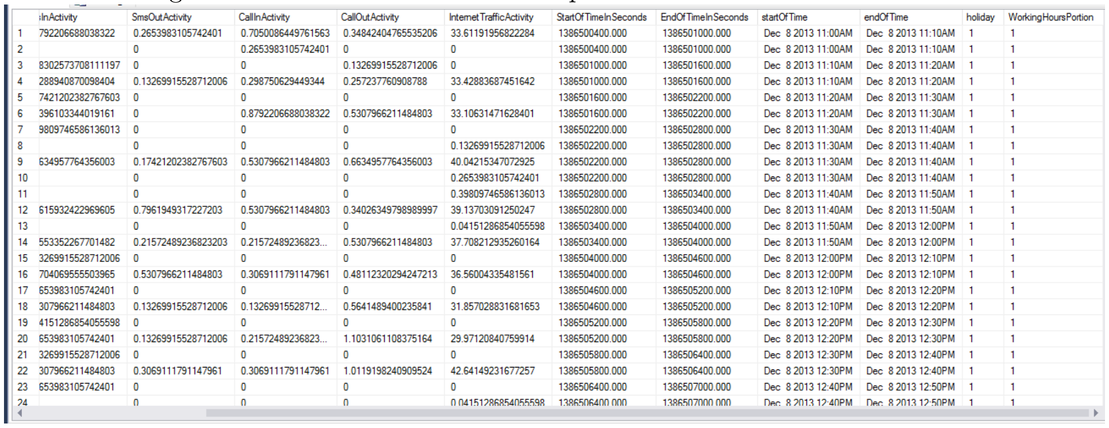
The above figures show the hierarchy of Vodafone dataset stored in JSON format in Firebase Realtime Database. "jsonFile" node is the parent for Vodafone dataset. This node contains children nodes which correspond to columns' names found in the dataset. Each node that corresponds to a certain column contains other children nodes that corresponds to column values. The hierarchy is three-level deep.

4.1.2 Telecom Italia Dataset Sql Server Database



	SquareID	TimeInterval	CountryCode	SmsInActivity	SmsOutActivity	CallInActivity	CallOutActivity	InternetTrafficActivity	StartOfTimeInSeconds	EndOfTimeInSeconds	startOfTime	endO
1	1841	1386500400000	39	0.8792206688038322	0.2653983105742401	0.7050086449761563	0.34842404765535206	33.61191956822284	1386500400.000	1386501000.000	Dec 8 2013 11:00AM	Dec
2	1841	1386500400000	44	0	0	0.2653983105742401	0	0	1386500400.000	1386501000.000	Dec 8 2013 11:00AM	Dec
3	1841	1386501000000	0	0.08302573708111197	0	0	0.13269915528712006	0	1386501000.000	1386501600.000	Dec 8 2013 11:10AM	Dec
4	1841	1386501000000	39	0.9288940870098404	0.13269915528712006	0.298750629449344	0.257237760908788	33.42883687451642	1386501000.000	1386501600.000	Dec 8 2013 11:10AM	Dec
5	1841	1386501600000	0	0.17421202382767603	0	0	0	0	1386501600.000	1386502200.000	Dec 8 2013 11:20AM	Dec
6	1841	1386501600000	39	0.4396103344019161	0	0.8792206688038322	0.5307966211484803	33.10631471628401	1386501600.000	1386502200.000	Dec 8 2013 11:20AM	Dec
7	1841	1386502200000	0	0.39809746586136013	0	0	0	0	1386502200.000	1386502800.000	Dec 8 2013 11:30AM	Dec
8	1841	1386502200000	358	0	0	0	0	0.13269915528712006	1386502200.000	1386502800.000	Dec 8 2013 11:30AM	Dec
9	1841	1386502200000	39	0.6634957764356003	0.17421202382767603	0.5307966211484803	0.6634957764356003	40.04215347072925	1386502200.000	1386502800.000	Dec 8 2013 11:30AM	Dec
10	1841	1386502200000	49	0	0	0	0	0.2653983105742401	1386502200.000	1386502800.000	Dec 8 2013 11:30AM	Dec
11	1841	1386502800000	358	0	0	0	0	0.39809746586136013	1386502800.000	1386503400.000	Dec 8 2013 11:40AM	Dec
12	1841	1386502800000	39	1.0615932422969605	0.7961949317227203	0.5307966211484803	0.34026349798989997	39.13703091250247	1386502800.000	1386503400.000	Dec 8 2013 11:40AM	Dec
13	1841	1386503400000	1	0	0	0	0	0.04151286854055598	1386503400.000	1386504000.000	Dec 8 2013 11:50AM	Dec
14	1841	1386503400000	39	0.6553352267701482	0.21572489236823203	0.21572489236823...	0.5307966211484803	37.708212935260164	1386503400.000	1386504000.000	Dec 8 2013 11:50AM	Dec
15	1841	1386504000000	0	0.13269915528712006	0	0	0	0	1386504000.000	1386504600.000	Dec 8 2013 12:00PM	Dec
16	1841	1386504000000	39	0.9704069555503965	0.5307966211484803	0.3069111791147961	0.48112320294247213	36.56004335481561	1386504000.000	1386504600.000	Dec 8 2013 12:00PM	Dec
17	1841	1386504600000	0	0.2653983105742401	0	0	0	0	1386504600.000	1386505200.000	Dec 8 2013 12:10PM	Dec
18	1841	1386504600000	39	0.5307966211484803	0.13269915528712006	0.13269915528712...	0.5641489400235841	31.857028831681653	1386504600.000	1386505200.000	Dec 8 2013 12:10PM	Dec
19	1841	1386505200000	0	0.04151286854055598	0	0	0	0	1386505200.000	1386505800.000	Dec 8 2013 12:20PM	Dec
20	1841	1386505200000	39	0.2653983105742401	0.13269915528712006	0.21572489236823...	1.1031061108375164	29.97120840759914	1386505200.000	1386505800.000	Dec 8 2013 12:20PM	Dec
21	1841	1386505800000	0	0.13269915528712006	0	0	0	0	1386505800.000	1386506400.000	Dec 8 2013 12:30PM	Dec
22	1841	1386505800000	39	0.5307966211484803	0.3069111791147961	0.3069111791147961	1.0119198240909524	42.64149231677257	1386505800.000	1386506400.000	Dec 8 2013 12:30PM	Dec
23	1841	1386506400000	0	0.2653983105742401	0	0	0	0	1386506400.000	1386507000.000	Dec 8 2013 12:40PM	Dec
24	1841	1386506400000	1	0	0	0	0	0.04151286854055598	1386506400.000	1386507000.000	Dec 8 2013 12:40PM	Dec

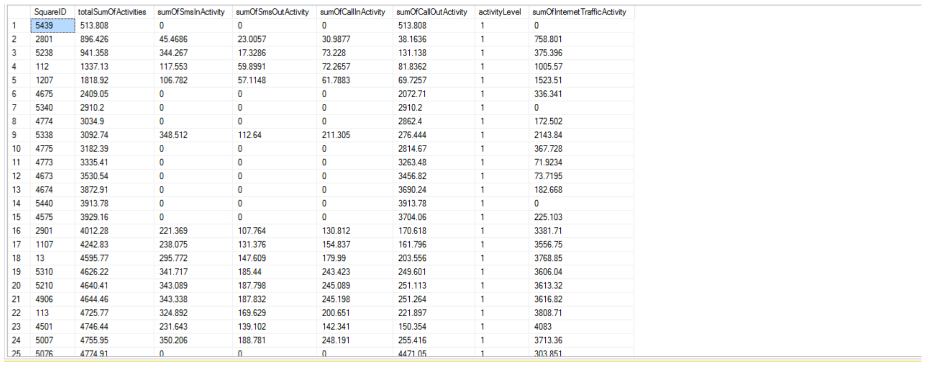
Figure 33: Telecom Italia Dataset Sql Server Database



	InActivity	SmsOutActivity	CallInActivity	CallOutActivity	InternetTrafficActivity	StartOfTimeInSeconds	EndOfTimeInSeconds	startOfTime	endOfTime	holiday	WorkingHoursPortion
1	792206688038322	0.2653983105742401	0.7050086449761563	0.34842404765535206	33.61191956822284	1386500400.000	1386501000.000	Dec 8 2013 11:00AM	Dec 8 2013 11:10AM	1	1
2		0	0.2653983105742401	0	0	1386500400.000	1386501000.000	Dec 8 2013 11:00AM	Dec 8 2013 11:10AM	1	1
3	8302573708111197	0	0	0.13269915528712006	0	1386501000.000	1386501600.000	Dec 8 2013 11:10AM	Dec 8 2013 11:20AM	1	1
4	288940870098404	0.13269915528712006	0.298750629449344	0.257237760908788	33.42883687451642	1386501000.000	1386501600.000	Dec 8 2013 11:10AM	Dec 8 2013 11:20AM	1	1
5	7421202382767603	0	0	0	0	1386501600.000	1386502200.000	Dec 8 2013 11:20AM	Dec 8 2013 11:30AM	1	1
6	396103344019161	0	0.8792206688038322	0.5307966211484803	33.10631471628401	1386501600.000	1386502200.000	Dec 8 2013 11:20AM	Dec 8 2013 11:30AM	1	1
7	9809746586136013	0	0	0	0	1386502200.000	1386502800.000	Dec 8 2013 11:30AM	Dec 8 2013 11:40AM	1	1
8		0	0	0	0.13269915528712006	1386502200.000	1386502800.000	Dec 8 2013 11:30AM	Dec 8 2013 11:40AM	1	1
9	634957764356003	0.17421202382767603	0.5307966211484803	0.6634957764356003	40.04215347072925	1386502200.000	1386502800.000	Dec 8 2013 11:30AM	Dec 8 2013 11:40AM	1	1
10		0	0	0	0.2653983105742401	1386502200.000	1386502800.000	Dec 8 2013 11:30AM	Dec 8 2013 11:40AM	1	1
11		0	0	0	0.39809746586136013	1386502800.000	1386503400.000	Dec 8 2013 11:40AM	Dec 8 2013 11:50AM	1	1
12	615932422969605	0.7961949317227203	0.5307966211484803	0.34026349798989997	39.13703091250247	1386502800.000	1386503400.000	Dec 8 2013 11:40AM	Dec 8 2013 11:50AM	1	1
13		0	0	0	0.04151286854055598	1386503400.000	1386504000.000	Dec 8 2013 11:50AM	Dec 8 2013 12:00PM	1	1
14	553352267701482	0.21572489236823203	0.21572489236823...	0.5307966211484803	37.708212935260164	1386503400.000	1386504000.000	Dec 8 2013 11:50AM	Dec 8 2013 12:00PM	1	1
15	3269915528712006	0	0	0	0	1386504000.000	1386504600.000	Dec 8 2013 12:00PM	Dec 8 2013 12:10PM	1	1
16	704069555503965	0.5307966211484803	0.3069111791147961	0.48112320294247213	36.56004335481561	1386504000.000	1386504600.000	Dec 8 2013 12:00PM	Dec 8 2013 12:10PM	1	1
17	653983105742401	0	0	0	0	1386504600.000	1386505200.000	Dec 8 2013 12:10PM	Dec 8 2013 12:20PM	1	1
18	307966211484803	0.13269915528712006	0.13269915528712...	0.5641489400235841	31.857028831681653	1386504600.000	1386505200.000	Dec 8 2013 12:10PM	Dec 8 2013 12:20PM	1	1
19	4151286854055598	0	0	0	0	1386505200.000	1386505800.000	Dec 8 2013 12:20PM	Dec 8 2013 12:30PM	1	1
20	653983105742401	0.13269915528712006	0.21572489236823...	1.1031061108375164	29.97120840759914	1386505200.000	1386505800.000	Dec 8 2013 12:20PM	Dec 8 2013 12:30PM	1	1
21	3269915528712006	0	0	0	0	1386505800.000	1386506400.000	Dec 8 2013 12:30PM	Dec 8 2013 12:40PM	1	1
22	307966211484803	0.3069111791147961	0.3069111791147961	1.0119198240909524	42.64149231677257	1386505800.000	1386506400.000	Dec 8 2013 12:30PM	Dec 8 2013 12:40PM	1	1
23	653983105742401	0	0	0	0	1386506400.000	1386507000.000	Dec 8 2013 12:40PM	Dec 8 2013 12:50PM	1	1
24		0	0	0	0.04151286854055598	1386506400.000	1386507000.000	Dec 8 2013 12:40PM	Dec 8 2013 12:50PM	1	1

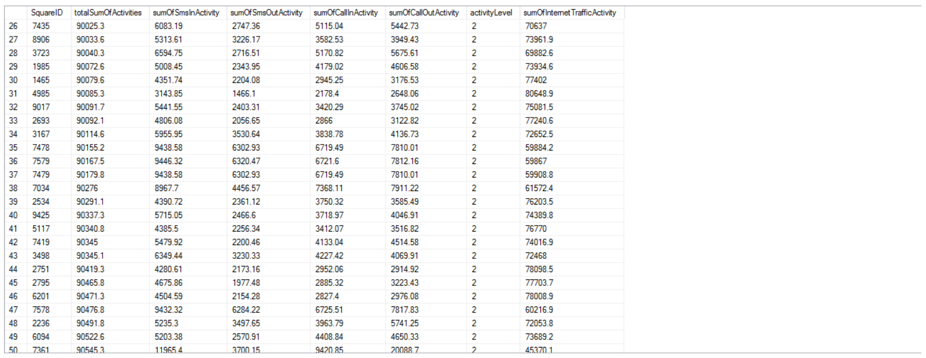
Figure 34: Telecom Italia Dataset Sql Server Database

The above figures are the dataset of telecom italia stored in SQL Server Database. This database consists of 14 columns and more than 90 million records for milano city in december 2013. 8 columns are the default columns in the dataset, and the rest 6 columns we added them to help us in our analysis which are start of time in seconds, end of time in seconds, start of time, end of time, holiday and working hours.



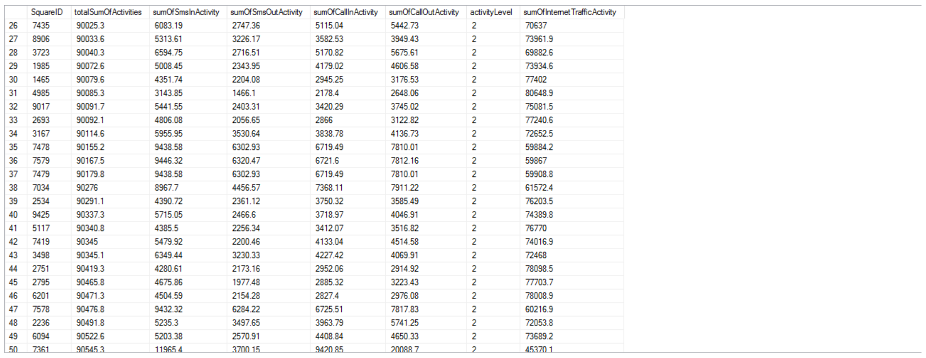
	SquareID	totalSumOfActivities	sumOfSmsInActivity	sumOfSmsOutActivity	sumOfCallInActivity	sumOfCallOutActivity	activityLevel	sumOfInternetTrafficActivity
1	5439	513.808	0	0	0	513.808	1	0
2	2801	896.426	45.4686	23.0057	30.9877	38.1636	1	758.801
3	5238	941.358	344.267	17.3286	73.228	131.138	1	375.396
4	112	1337.13	117.553	59.8991	72.2657	81.8362	1	1005.57
5	1207	1818.92	106.782	57.1148	61.7883	69.7257	1	1523.51
6	4675	2409.05	0	0	0	2072.71	1	336.341
7	5340	2910.2	0	0	0	2910.2	1	0
8	4774	3034.9	0	0	0	2862.4	1	172.502
9	5338	3092.74	348.512	112.64	211.305	276.444	1	2143.84
10	4775	3182.39	0	0	0	2814.67	1	367.728
11	4773	3335.41	0	0	0	3263.48	1	71.9234
12	4673	3530.54	0	0	0	3456.82	1	73.7195
13	4674	3872.91	0	0	0	3690.24	1	182.668
14	5440	3913.78	0	0	0	3913.78	1	0
15	4575	3929.16	0	0	0	3704.06	1	225.103
16	2901	4012.28	221.369	107.764	130.812	170.618	1	3381.71
17	1107	4242.83	238.075	131.376	154.837	161.796	1	3556.75
18	13	4595.77	295.772	147.609	179.99	203.556	1	3768.85
19	5310	4626.22	341.717	185.44	243.423	249.601	1	3606.04
20	5210	4640.41	343.089	187.798	245.089	251.113	1	3613.32
21	4906	4644.46	343.338	187.832	245.198	251.264	1	3616.82
22	113	4725.77	324.892	169.629	200.651	221.897	1	3808.71
23	4501	4746.44	231.643	139.102	142.341	150.354	1	4083
24	5007	4755.95	350.206	188.781	248.191	255.416	1	3713.36
25	5076	4774.91	0	0	0	4471.05	1	303.851

Figure 35: Telecom Italia Dataset Sql Server Database selected 75 square id



	SquareID	totalSumOfActivities	sumOfSmsInActivity	sumOfSmsOutActivity	sumOfCallInActivity	sumOfCallOutActivity	activityLevel	sumOfInternetTrafficActivity
26	7435	90025.3	6083.19	2747.36	5115.04	5442.73	2	70637
27	8906	90033.6	5313.61	3226.17	3582.53	3949.43	2	73961.9
28	3723	90040.3	6594.75	2716.51	5170.82	5675.61	2	69882.6
29	1985	90072.6	5008.45	2343.95	4179.02	4606.58	2	73934.6
30	1465	90079.6	4351.74	2204.08	2945.25	3176.53	2	77402
31	4985	90085.3	3143.85	1466.1	2178.4	2648.06	2	80648.9
32	9017	90091.7	5441.55	2403.31	3420.29	3745.02	2	75081.5
33	2693	90092.1	4806.08	2056.65	2866	3122.82	2	77240.6
34	3167	90114.6	5955.95	3530.64	3838.78	4136.73	2	72652.5
35	7478	90155.2	9438.58	6302.93	6719.49	7810.01	2	59884.2
36	7579	90167.5	9446.32	6320.47	6721.6	7812.16	2	59867
37	7479	90179.8	9438.58	6302.93	6719.49	7810.01	2	59908.8
38	7034	90276	8967.7	4456.57	7368.11	7911.22	2	61572.4
39	2534	90291.1	4390.72	2361.12	3750.32	3585.49	2	76203.5
40	9425	90337.3	5715.05	2466.6	3718.97	4046.91	2	74389.8
41	5117	90340.8	4385.5	2256.34	3412.07	3516.82	2	76770
42	7419	90345	5479.92	2200.46	4133.04	4514.58	2	74016.9
43	3498	90345.1	6349.44	3230.33	4227.42	4069.91	2	72468
44	2751	90419.3	4280.61	2173.16	2952.06	2914.92	2	78098.5
45	2795	90465.8	4675.86	1977.48	2885.32	3223.43	2	77703.7
46	6201	90471.3	4504.59	2154.28	2827.4	2976.08	2	78008.9
47	7578	90476.8	9432.32	6284.22	6725.51	7817.83	2	60216.9
48	2236	90491.8	5235.3	3497.65	3963.79	5741.25	2	72053.8
49	6094	90522.6	5203.38	2570.91	4408.84	4650.33	2	73689.2
50	7361	90545.3	11965.4	3700.15	9420.85	20088.7	2	45370.1

Figure 36: Telecom Italia Dataset Sql Server Database selected 75 square id



	SquareID	totalSumOfActivities	sumOfSmsInActivity	sumOfSmsOutActivity	sumOfCallInActivity	sumOfCallOutActivity	activityLevel	sumOfInternetTrafficActivity
26	7435	90025.3	6083.19	2747.36	5115.04	5442.73	2	70637
27	8906	90033.6	5313.61	3226.17	3582.53	3949.43	2	73961.9
28	3723	90040.3	6594.75	2716.51	5170.82	5675.61	2	69882.6
29	1985	90072.6	5008.45	2343.95	4179.02	4606.58	2	73934.6
30	1465	90079.6	4351.74	2204.08	2945.25	3176.53	2	77402
31	4985	90085.3	3143.85	1466.1	2178.4	2648.06	2	80648.9
32	9017	90091.7	5441.55	2403.31	3420.29	3745.02	2	75081.5
33	2693	90092.1	4806.08	2056.65	2866	3122.82	2	77240.6
34	3167	90114.6	5955.95	3530.64	3838.78	4136.73	2	72652.5
35	7478	90155.2	9438.58	6302.93	6719.49	7810.01	2	59884.2
36	7579	90167.5	9446.32	6320.47	6721.6	7812.16	2	59867
37	7479	90179.8	9438.58	6302.93	6719.49	7810.01	2	59908.8
38	7034	90276	8967.7	4456.57	7368.11	7911.22	2	61572.4
39	2534	90291.1	4390.72	2361.12	3750.32	3585.49	2	76203.5
40	9425	90337.3	5715.05	2466.6	3718.97	4046.91	2	74389.8
41	5117	90340.8	4385.5	2256.34	3412.07	3516.82	2	76770
42	7419	90345	5479.92	2200.46	4133.04	4514.58	2	74016.9
43	3498	90345.1	6349.44	3230.33	4227.42	4069.91	2	72468
44	2751	90419.3	4280.61	2173.16	2952.06	2914.92	2	78098.5
45	2795	90465.8	4675.86	1977.48	2885.32	3223.43	2	77703.7
46	6201	90471.3	4504.59	2154.28	2827.4	2976.08	2	78008.9
47	7578	90476.8	9432.32	6284.22	6725.51	7817.83	2	60216.9
48	2236	90491.8	5235.3	3497.65	3963.79	5741.25	2	72053.8
49	6094	90522.6	5203.38	2570.91	4408.84	4650.33	2	73689.2
50	7361	90545.3	11965.4	3700.15	9420.85	20088.7	2	45370.1

Figure 37: Telecom Italia Dataset Sql Server Database selected 75 square id

The above three figures for the telecom italia dataset are for selected 75 square id with their total activites, sum of each activity and their activity level. The selected 75 square id are selected according to 25 low activity level, 25 moderate activity level and 25 high activity level. The main purpose of this table is to determine the activity level of each square id.

4.2 Data Dictionary

Telecom Italia dataset:

Attribute Name	Description
Square id	identification string of a given square of Milan GRID
Time Interval	start interval time expressed in milliseconds. The end interval time can be obtained by adding 600,000 milliseconds (10 min) to this value
SMS-in activity	activity proportional to the amount of received SMSs inside a given Square id and during a given Time interval. The SMSs are sent from the nation identified by the Country code
SMS-out activity	activity proportional to the amount of sent SMSs inside a given Square id during a given Time interval. The SMSs are received in the nation identified by the Country code
Call-in activity	activity proportional to the amount of received calls inside the Square id during a given Time interval. The calls are issued from the nation identified by the Country code
Call-out activity	activity proportional to the amount of issued calls inside a given Square id during a given Time interval. The calls are received in the nation identified by the Country code
Internet traffic activity	number of CDRs generated inside a given Square id during a given Time interval. The Internet traffic is initiated from the nation identified by the Country code
Country code	the phone country code of the nation
Start Of Time In Seconds	time interval converted from milliseconds to seconds to get the start time in seconds
End Of Time In Seconds	time interval plus 600,000 milliseconds, converted to seconds to set the end time in seconds
Start Of Time	start of time converted to hours and minutes, specifying the date (Month / day / year) of each activity
End Of Time	end of time converted to hours and minutes, specifying the date (Month / day / year) of each activity
Holiday	specifying the days which are weekends in italy (saturday and sunday) with also the public european holidays. The holidays specified in the column as 1.
Working Hours Portion	this column specifying the working hours portion as from 8:00 am to 2:00 pm, it is set to 1, from 2:00 pm to 6:00 pm, it is set to 2 and from 6:00 pm to 12:00 am, it is set to 3
total sum of activities	total summation of all activities for each square id
sum of sms in activity	summation of sms in activity for each square id
sum of sms out activity	summation of sms out activity for each square id
sum of call in activity	summation of call in activity for each square id
sum of call out activity	summation of call out activity for each square id
sum of internet traffic activity	summation of internet traffic activity for each square id
activity level	specifying according to sum of each activity with the total sum of activites

4.2.1 File, File Option , File Value and Option Tables

In these four tables, we used the entity attribute value design to be compatible with Vodafone data sets (Capacity Monitoring Tool) samples and Telecom Italia (Telecommunication Activity) . We used the entity attribute value because the structure of the data sets is ambiguous and unclear and to avoid the null values.

4.2.2 User, User Type , User URL and URL Tables

These tables designed for storing user information (administrator and capacity management team) as username, password and email for Vodafone Company. In the table User Type , the user are defined by their type. According to this type, the user can be directed to specific web pages.

4.2.3 notification, notificationType Tables

These tables are designed for notifying users if there are a problem as for Vodafone Company, the capacity management team will be notified if there are a overhead in nodes capacity.

4.2.4 dashboard Table

This table is designed for collecting all types of charts and gauges, and a foreign key in the table of userType, as according to the user type, each user has their own charts and gauges. Moreover, a foreign key in the table report as while producing the reports, charts and gauges will be also produced and summaries things to the users.

4.2.5 report Table

This table is designed to make reports and collect all the information in one place like user information, file information, node information, time, date and displaying dashboards (charts and gauges)

4.2.6 node Table

This table is designed to collect all nodes' information.

5 Component Design

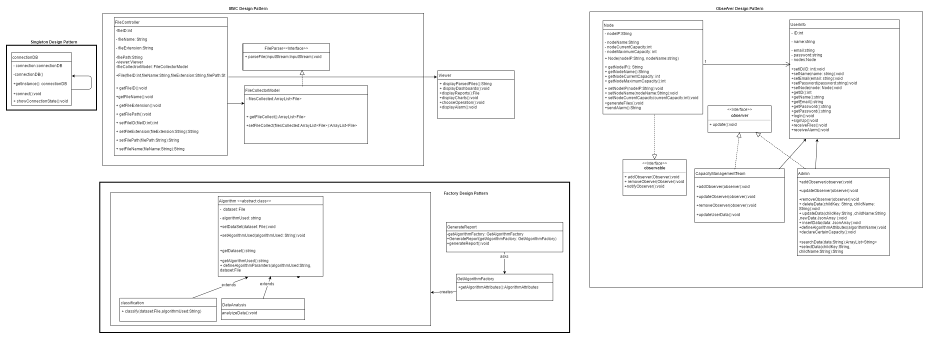

Singleton Design Pattern
connectionDB
- connection:connectionDB
-connectionDB()
+getInstance(): connectionDB
+connect():void
+ showConnectionState():void
MVC Design Pattern
FileController
-fileID:int
- fileName: String
- fileExtension:String
-filePath:String
-viewer:Viewer
-fileCollectrorModel: FileCollectorModel
+File(fileID:int,fileName:String,fileExtension:String,filePath:St
+ getFileID():void
+getFileName():void
+ getFileExtension():void
+ getFilePath():void
+ setFileID(fileID:int):int
+ setFileExtension(fileExtension:String):String
+ setFilePath(filePath:String):String
+ setFileName(fileName:String):String
FileParser<<Interface>>
+ parseFile(inputStream:InputStream):void
FileCollectorModel
- filesCollected:ArrayList<File>
+ getFileCollect():ArrayList<File>
+setFileCollect(filesCollected:ArrayList<File>):ArrayList<File>
Viewer
+ displayParsedFiles():String
+ displayDashboards():void
+displayReports():File
+displayCharts():void
+chooseOperation():void
+displayAlarm():void
Factory Design Pattern
Algorithm <<abstract class>>
- dataset: File
- algorithmUsed: string
+setDataSet(dataset: File):void
+setAlgorithmUsed(algorithmUsed: String):void
+getDataset():string
+getAlgorithmUsed():string
+ defineAlgorithmParamters(algorithmUsed:String, dataset:File
extends
extends
classification
+ classify(dataset:File,algorithmUsed:String)
DataAnalysis
analyizeData():void
GenerateReport
-getAlgorithmFactory: GetAlgorithmFactory
+GenerateReport(getAlgorithmFactory: GetAlgorithmFactory)
+generateReport():void
asks
GetAlgorithmFactory
+getAlgorithmAttributes():AlgorithmAttributes
creates
Observer Design Pattern
Node
- nodeIP:String
- nodeName:String
- nodeCurrentCapacity:int
- nodeMaximumCapacity: int
+ Node(nodeIP:String, nodeName:string)
+ getNodeIP(): String
+ getNodeName():String
+ getNodeCurrentCapacity: int
+ getNodeMaximumCapacity():int
+ setNodeIP(nodeIP:String):void
+ setNodeName(nodeName:String):void
+ setNodeCurrentCapacity(currentCapacity:int):void
+generateFiles();void
+sendAlarm():String
1
UserInfo
- ID:int
- name:string
- email:string
- password:string
- nodes:Node
+setID(ID: int):void
+setName(name: string):void
+setEmail(email: string):void
+setPassword(password:string):void
+setNode(node: Node):void
+getID():int
+getName():string
+getEmail():string
+getPassword():string
+getPassword():string
+login():void
+signUp():void
+receiveFiles():void
+receiveAlarm():void
<<Interface>>
observer
+ update():void
<<Interface>>
observable
+ addObserver(Observer):void
+ removeObserver(Observer):void
+notifyObserver():void
CapacityManagementTeam
+addObserver(observer):void
+updateObserver(observer):void
+removeObserver(observer):void
+updateUserData():void
Admin
+addObserver(observer):void
+updateObserver(observer):void
+removeObserver(observer):void
+ deleteData(childKey: String, childName: String):void
+ updateData(childKey:String ,childName:String ,newData:JsonArray):void
+ insertData(data: JsonArray):void
+defineAlgorithmAttributes(algorithmName):void
+declareCertainCapacity():void
+searchData(data:String):ArrayList<String>
+selectData(childKey:String, childName:String):String

Figure 38: Class Diagram

5.0.1 connectionDB

This class is a concrete class.

5.0.2 List of Superclasses:

No super classes for this class.

5.0.3 List of Subclasses:

No sub-classes for this class.

5.0.4 Purpose:

The basic purpose of this class is to establish connection to Firebase database. It follows the Singleton design pattern since the connection to the database is only required once.

5.0.5 Collaborations:

No collaborations are needed.

5.0.6 Attributes:

connection: is an instance of the connectionDB class.

5.0.7 Operations

: getInstance(): is a static method that returns one and only one object of this class.

connect(): has no return value. Its purpose is to connect to the database.

showConnectionState(): has no return type. Its purpose is to show the connection to the database state.

5.0.8 Constraints:

Only one object of this class can be instantiated and gotten through the static function getInstance().

5.0.9 FileCollectorModel

This class is a concrete class.

5.0.10 List of Superclasses:

No super classes for this class.

5.0.11 List of Subclasses:

No sub-classes for this class.

5.0.12 Purpose:

The basic purpose of this class is to collect files that vary in formats from the network nodes.

5.0.13 Collaborations:

This class must interact with FileController class in-order to achieve the required functionalities of this class. It also has to implement FileParser interface for the parsing files process.

5.0.14 Attributes:

filesCollected: is of type ArrayList¡File¿ that stores the collected files from the network nodes.

5.0.15 Operations

: getFilesCollected(): is a method that returns an array list of type File.

setFilesCollected(filesCollected: ArayList¡File): assigns the files that are collected from the network nodes to filesCollected variable.

5.0.16 Constraints:

No constraints.

5.0.17 FileController

This class is a concrete class.

5.0.18 List of Superclasses:

No super classes for this class.

5.0.19 List of Subclasses:

No sub-classes for this class.

5.0.20 Purpose:

The basic purpose of this class is to link between FileCollector-Model class, and Viewer class.

5.0.21 Collaborations:

This class must interact with FileController class, and and Viewer class in-order to achieve the required functionalities of the linked classes mentioned earlier.

5.0.22 Attributes:

fileName, fileExtention, filePath: are of type String.
fileID: is of type integer
viewer: is an object of type Viewer class.
filesCollectedModel: is an object of type FilesCollectedModel class.

5.0.23 Operations

: getFileID(): returns an integer value that corresponds to the ID of the file.
getFileName(): returns a String value that corresponds to the name of the file.
getFileExtension(): returns a String value that corresponds to the extension/ format of the file.
getFilePath(): returns a String value that corresponds to the path of the file.
setFileID(fileID: int): assigns an integer value for the fileID variable.
setFileName(fileName: String): assigns a String value for the fileName variable.
setFilePath(filePath: String): assigns a String value for the filePath variable.

setFileExtension(fileExtension: String): assigns a String value for the fileExtension variable.

5.0.24 Constraints:

No constraints.

5.0.25 FileParser

This is an interface.

5.0.26 List of Superclasses:

No super classes for this class.

5.0.27 List of Subclasses:

No sub-classes for this class.

5.0.28 Purpose:

The basic purpose of this interface is to provide common functionalities but allows different implementations for the parsing files process.

5.0.29 Collaborations:

FileCollectorModel class has to implement this interface in-order to perform the parsing files process.

5.0.30 Attributes:

No attributes.

5.0.31 Operations

: parseFile(inputStream :InputStream): has no return value. Its purpose is to parse files with different formats. Its implementation will be in the FileCollectorModel class that implements this interface.

5.0.32 Constraints:

Only one object of this class can be instantiated and gotten through the static function getInstance().

5.0.33 Viewer

This class is a concrete class.

5.0.34 List of Superclasses:

No super classes for this class.

5.0.35 List of Subclasses:

No sub-classes for this class.

5.0.36 Purpose:

The basic purpose of this class is to display the data coming from the FileCollectorModel class.

5.0.37 Collaborations:

This class must interact with FileController class in-order to achieve the required functionalities.

5.0.38 Attributes:

No attributes.

5.0.39 Operations

: displayParsedFiles(): is a method that returns a String value for the parsed files.
displayDashboards(): has no return value. Its purpose is to display dashboards that contain a summary of the data.
displatCharts(): has no return value. Its purpose is to display charts for the data.
displayAlarm(): has no return type. Its purpose is to display alarms received when capacity issues are about to occur.
displayReports(): has no return value. Its purpose is to display reports generated that contain summary about the results of performing operations on data such as classification, clustering, and data analysis.
chooseOperation(): has no return value. Its purpose is to let the user choose the operation he/she wishes to perform on the data such as classification, clustering, and data analysis.

5.0.40 Constraints:

No constraints.

5.0.41 Node

This class is a concrete class.

5.0.42 List of Superclasses:

No super classes for this class.

5.0.43 List of Subclasses:

No sub-classes for this class.

5.0.44 Purpose:

The basic purpose of this class is to contain data about each node such as IP, name, current capacity, and maximum capacity.

5.0.45 Collaborations:

This class must interact with UserInfo class and implement observable interface.

5.0.46 Attributes:

nodeIP, and nodeName: are of type String.

nodeCurrentCapacity, and nodeMaximumCapacity: are of type integer.

5.0.47 Operations

: getNodeIP(): returns a String value that corresponds to the node IP.

getNodeName(): returns a String value that corresponds to the node name.

getNodeCurrentCapacity(): returns an integer value that corresponds to the node current capacity.

getNodeMaximumCapacity(): returns an integer value that corresponds to the node maximum capacity.

setNodeIP(nodeIP:String):assigns a String value to nodeIP variable.

setNodeName(nodeName:String):assigns a String value to nodeName variable.

setNodeCurrentCapacity(nodeCurrentCapacity:String):assigns an integer value to

nodeCurrentCapacity variable.

generateFiles():generates files for each node containing some data.

sendAlarm(): returns a String value containing the alarm message if any capacity issues are about to occur.

5.0.48 Constraints:

No constraints.

5.0.49 Admin

This class is a concrete class.

5.0.50 List of Superclasses:

It extends from super-class ”UserInfo”.

5.0.51 List of Subclasses:

No sub-classes for this class.

5.0.52 Purpose:

The basic purpose of this class is to give privileges and operations for the admin to perform operations such as accessing database and managing it.

5.0.53 Collaborations:

This class must extend from UserInfo class and implement observer interface.

5.0.54 Attributes:

Attributes are the same as found in the super-class UserInfo.

5.0.55 Operations

: Methods are the same as found in super-class UserInfo but in addition to the following methods:

deleteData(childKey: String, childName:String): is a method with no return value. Its purpose is to give the admin privileges to delete data from the database. In-order to perform the deletion operation, some parameters are required: childKey is the the key in the database, and childName is the name of the child in database. Both can be used to delete a certain data or just one of them.

updateData(childKey:String, childName:String, newData:JSONArray): has no return value. Its purpose is to give privileges to the admin to update data in the database through providing certain parameters: childKey, childName, and newData.

insertData(newData:JSONArray): has no return value. Its purpose is to give privileges to the admin to insert data into the database through passing the new data in JSONArray format as a parameter.

defineAlgorithmAttributes(algorithmName:String): has no return value. Its purpose is to give the admin privileges to define and set default attribute values for a certain algorithm.

declareCertainCapacity():has no return value. Its purpose is to give the admin privileges to declare and set the maximum capacities for the network nodes.

searchData(data:String): returns an ArrayList of type String. Its purpose is to search for data in the database through passing a query as a parameter for the method.

selectData(childKey:String, childName:String): returns a String value. Its purpose is to select some data from the database through passing the childKey and childName as parameters to the method.

5.0.56 Constraints:

No constraints.

5.0.57 CapacityManagementTeam

This class is a concrete class.

5.0.58 List of Superclasses:

It extends from super-class ”UserInfo”.

5.0.59 List of Subclasses:

No sub-classes for this class.

5.0.60 Purpose:

The basic purpose of this class is to give privileges and operations for the capacity management team members to perform some operations..

5.0.61 Collaborations:

This class must extend from UserInfo class and implement observer interface.

5.0.62 Attributes:

Attributes are the same as found in the super-class UserInfo.

5.0.63 Operations

: Methods are the same as found in super-class UserInfo but in addition to the following methods:
updateUserData(): has no return value. Its purpose is to give privileges to each capacity management team member to update his/her information.

5.0.64 Constraints:

No constraints.

5.1 Classification, and DataAnalysis

These classes are sub-classes that extend from abstract class "Algorithm".

5.1.1 List of Superclasses:

There is a super class "Algorithm".

5.1.2 Purpose:

The basic purpose of these classes is to apply any of these algorithms on the data.

5.1.3 Collaborations:

No collaborations are needed. Attributes:

5.1.4 Operations

: classify(dataset: File, algorithmUsed:String) its the function that classify data in files.
analyzeData(): has no return value.

5.1.5 Constraints:

No constraints.

5.2 Algorithm

This class is an abstract class.

5.2.1 List of Superclasses:

No super classes for this class.

5.2.2 List of Subclasses:

subclasses: classification, and dataAnalysis

5.2.3 Purpose:

The basic purpose of this class is to choose an algorithm to classify or analyise data.

5.2.4 Collaborations:

No collaborations are needed. Attributes:

5.2.5 Operations

: setDataSet(dataset:File): has no return value, purpose to set/upload data file.

setAlgorithmUsed(algorithmUsed:String): has no return value, purpose to set an algorithm on data.

defineAlgorithmParameters(algorithmUsed:String, dataset:File): is a method that define type of algorithm for the file.

getDataSet: return files that has been uploaded.

getAlgorithmUsed: return algorithms that has been choosen.

5.2.6 Constraints:

No constraints.

5.3 UserInfo

This class is a super-class

5.3.1 Subclasses:

Sub-classes are : Admin and CapacityManagementTeam

5.3.2 Purpose:

The basic purpose of this class is to provide common attributes and methods to its sub-classes.

5.3.3 Collaborations:

This class is a super-class, both Admin, and CapacityManagementTeam extend from this class. Also, there is n association relationship between this class and Node class. Attributes: name, email, and password: are of type String representing user name, email, ad password.

id: is of type integer representing user ID.

nodes: is of type Node, representing the node that is monitored by a certain member in the capacity management team.

5.3.4 Operations

: getID(): returns an integer value that corresponds to the ID.

getName(): returns a String value that corresponds to the name.

getEmail(): returns a tring value that corresponds to the email.

getPassword(): returns a String value that corresponds to the password.

setID(id:int): assigns an integer value to variable ID.

setName(name:String): assigns a String value to variable name.

setEmail(email:String): assigns a String value to variable email.

setPassword(password:String): assigns a String value to variable password.

logIn(): has no return value. Its purpose is to handle the logging-in process.

signUp(): has no return value. Its purpose is to handle the signing-up process.

receiveFiles(): has no return value. Its purpose is to handle the process of receiving files.

receiveAlarm(): has no return value. Its purpose is to receive data about the alarm sent.

5.3.5 Constraints:

5.4 observable

This is an interface for handling the operation of add or delete observer and its notify.

5.4.1 Subclasses:

No subclasses.

5.4.2 Purpose:

The basic purpose of this interface is to add or delete observer and send notification to the observers.

5.4.3 Collaborations:

This interface interacts with node class which will implements the function inside the interface. Attributes: No attributes.

5.5 observer

This is an interface for updating the notifications.

5.5.1 Subclasses:

No subclasses.

5.5.2 Purpose:

The basic purpose of this interface is to update the notification.

5.5.3 Collaborations:

This interface interacts with capacity management team and admin classes which will implements the function inside the interface. Attributes: No attributes.

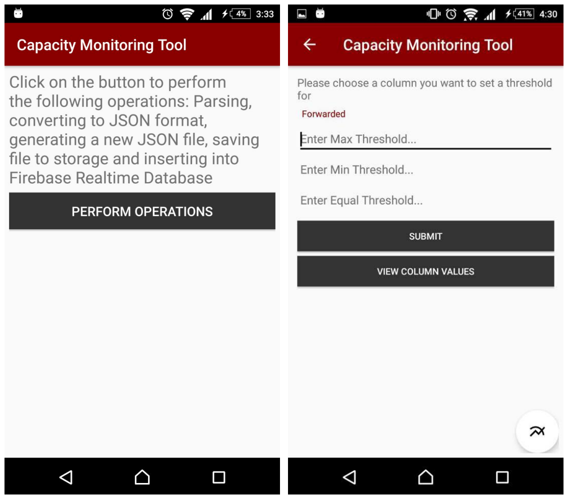
6 Human Interface Design

6.1 Overview of User Interface

Capacity monitoring tool allow user to upload/fetch file, then click on perform button which makes several operations without user knowledge as inserting into Firebase RealTime database, converting into JSON and parsing.Then the user could select a certain column and view its values.Also, the user could insert a maximum, minimum and equal threshold.Moreover, the user could display several types of dashboards (charts and gauges).

6.2 Screen Images

Android mobile application



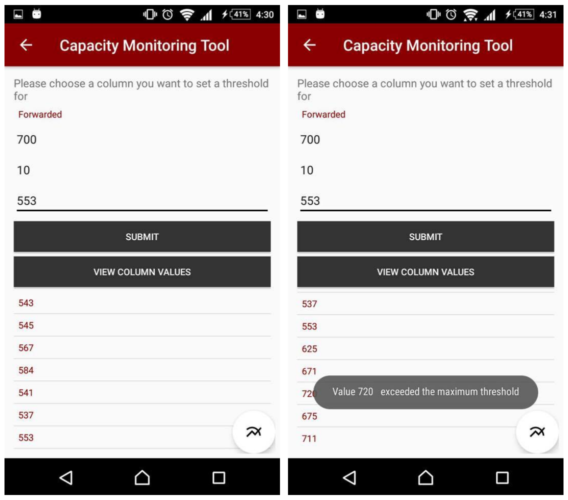
Click on the button to perform the following operations: Parsing, converting to JSON format, generating a new JSON file, saving file to storage and inserting into Firebase Realtime Database

Please choose a column you want to set a threshold for

Forwarded

Figure 39: Home Page / Set Threshold Page

The Home Page in the above figure is consist of a button to perform certain operations as parsing, converting the file to JSON formate, generate a new JSON file, saving file to storage and inserting into Firebase Realtime Database.Then when pressing the perform operations button, a new activity appears having three textbox to specify the maximum, minimum and equal threshold then a button to submit and set the threshold.Moreover you could choose any column in the dataset to set the threshold for it. Moreover, there is a button for graphing the results.



Please choose a column you want to set a threshold for

Forwarded

700

10

553

543
545
567
584
541
537
553

Please choose a column you want to set a threshold for

Forwarded

700

10

553

537
553
625
671
720
675
711

Value 720 exceeded the maximum threshold

Figure 40: Retrieve Values / Toast Threshold page

The above two activities are for retrieving the results for the specified threshold in a listview and toast for specifying the exceeded values.

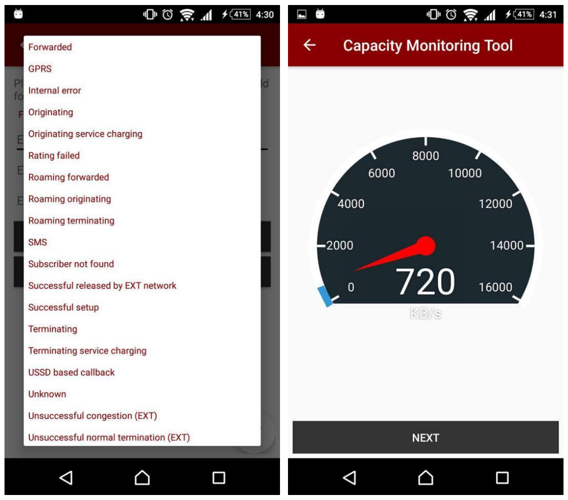

Forwarded
GPRS
Internal error
Originating
Originating service charging
Rating failed
Roaming forwarded
Roaming originating
Roaming terminating
SMS
Subscriber not found
Successful released by EXT network
Successful setup
Terminating
Terminating service charging
USSD based callback
Unknown
Unsuccessful congestion (EXT)
Unsuccessful normal termination (EXT)

Figure 41: Columns Name / Gauge

The activity on the left side is showing drop down list for the columns of the dataset. And for the activity which is on the right side, it is for displaying one of the graphs types which is gauge.

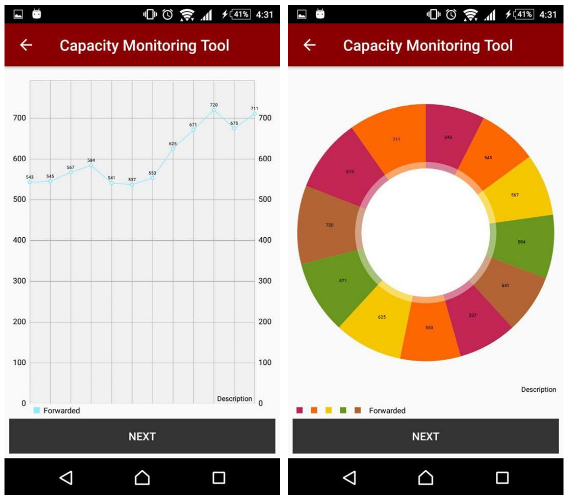
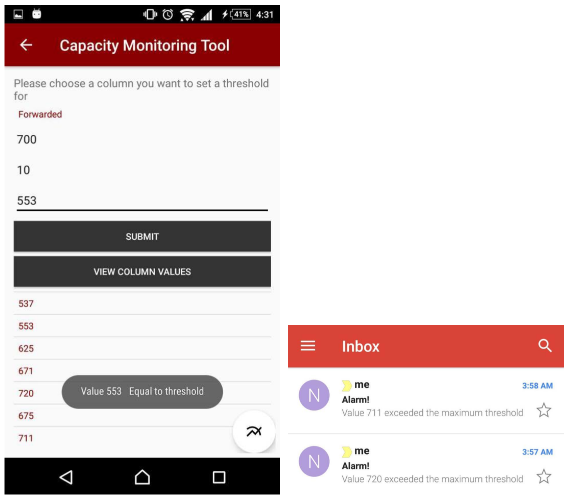


Figure 42: Line Chart / Pie Chart

These two activities displaying other types of charts which are line chart and pie chart



Capacity Monitoring Tool

Please choose a column you want to set a threshold for

Forwarded

700

10

553

537
553
625
671
720
675
711

Value 553 Equal to threshold

me 3:58 AM
Alarm!
Value 711 exceeded the maximum threshold

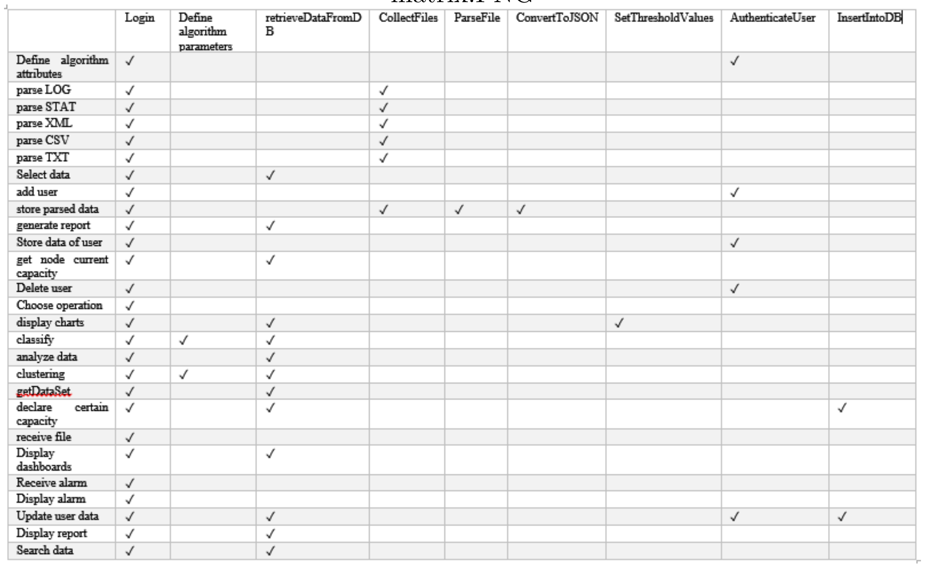
me 3:57 AM
Alarm!
Value 720 exceeded the maximum threshold

Figure 43: Toast With Values Exceeed page / Mail alert page

The activity on the left side displaying a toast for the exceeded values.And the right side for the mail alert for alarming the user.

7 Requirements Matrix

matrix.PNG



	Login	Define algorithm parameters	retrieveDataFromDB	CollectFiles	ParseFile	ConvertToJSON	SetThresholdValues	AuthenticateUser	InsertIntoDB
Define algorithm attributes	✓							✓	
parse LOG	✓			✓					
parse STAT	✓			✓					
parse XML	✓			✓					
parse CSV	✓			✓					
parse TXT	✓			✓					
Select data	✓		✓						
add user	✓							✓	
store parsed data	✓			✓	✓	✓			
generate report	✓		✓						
Store data of user	✓							✓	
get node current capacity	✓		✓						
Delete user	✓							✓	
Choose operation	✓								
display charts	✓		✓				✓		
classify	✓	✓	✓						
analyze data	✓		✓						
clustering	✓	✓	✓						
getDataSet	✓		✓						
declare certain capacity	✓		✓						✓
receive file	✓								
Display dashboards	✓		✓						
Receive alarm	✓								
Display alarm	✓								
Update user data	✓		✓					✓	✓
Display report	✓		✓						
Search data	✓		✓						

Figure 44: Requirement Matrix

8 References

1-https://dandelion.eu/datamine/open-big-data/
2-https://www.nature.com/articles/sdata201555data-records
3-https://www.kaggle.com/marcodena/mobile-phone-activity
4-http:www.gsma.com/newsroom/technical-documents
5-http:www.slideshare.net/yusufd/introduction-to-mobile-core-network-17667704
6-http:www.3gpp.org/specifications/79-specification-numbering
7-http:www.actuate.com/download/casestudy/Telecom-Hadoop-Case-Study.pdf
8-https:networks.nokia.com/solutions/performance-manager
9-https:www.youtube.com/watch?v=bFrvSt50VGk
10-https:aaltodoc.aalto.fi/bitstream/handle/123456789/4412/isbn9789512292837.pdf
11- https://www.csie.ntu.edu.tw/ cjlin/libsvm/
12-https://www.quora.com/What-is-the-difference-between-C-SVM-and-nu-SVM
13-http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/svmtoy3d/