Capacity Monitoring Tool

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

April 2, 2017

1 Introduction

1.1 Purpose of this Document

The purpose of this document is to present a detailed description of capacity monitoring tool. Including the purpose, features, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers.

1.2 Scope of this Document

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 a 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. 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

The proposed tool consists of two parts: a website and a 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 and inserts the needed data into Firebase database. Once parsing files is done, there are some operations that can be performed on the data extracted from

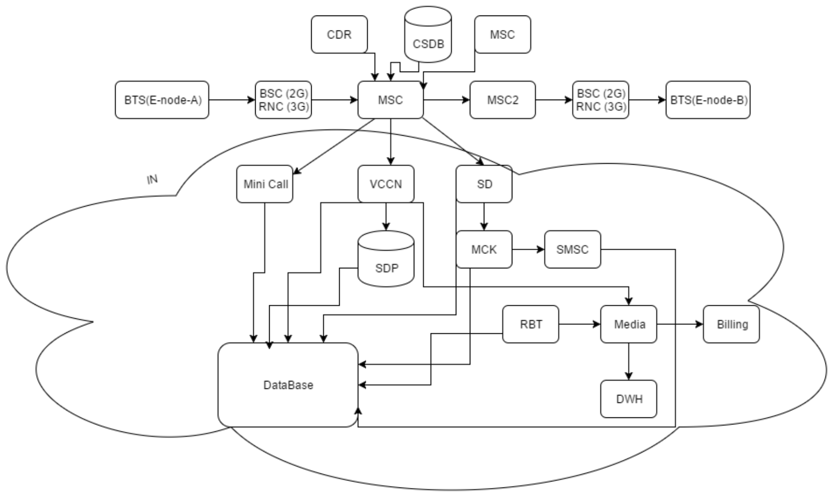


Figure 1: Vodafone Call Flow

these files such as the following: clustering, classification, and data analysis that include the following operations: average, sum, maximum and minimum to produce reports. Moreover to monitor the capacity of nodes, displayed dashboards of the capacity of nodes,are viewed for the capacity monitoring team. The mobile application will be developed for receiving alarms in case of the capacity of a certain node is about to reach its maximum capacity either via sending an SMS or an e-mail. In addition, the 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 files contain rows of data. The first row contains columns' names, the rest are the data that corresponds to each column.

The Call is Represented by:

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

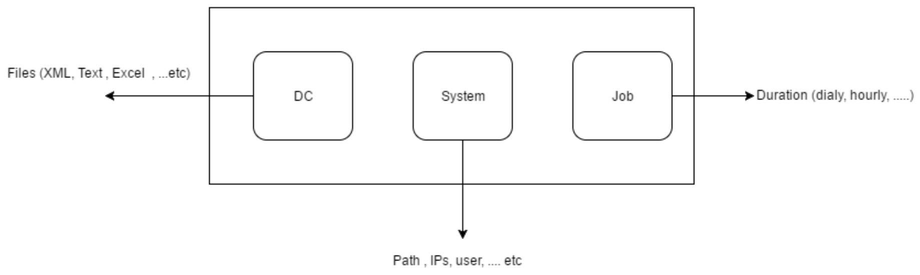

Files (XML, Text , Excel , ...etc)
DC
System
Job
Duration (dialy, hourly, ...)
Path , IPs, user, ... etc

Figure 2: SDP Diagram

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.

1.4 Business Context

The tool monitors the issue of the massive number of reports which are generated regularly. The purpose of generating reports is to make valuable offers that can benefit the business and help in guarantying a stable network performance by monitoring its resources.

2 General Description

2.1 Product Functions

The Capacity monitoring tool's functionalities will be collecting files from network nodes, parsing these files that vary in formats, clustering, classification, sending alarms/notifications, reviewing summaries and data analysis, which will be described in more details later in this document.

2.2 Similar System Information

There are similar systems for viewing the dashboards as Nokia and NSN company. A comparison is carried on between these systems and the proposed system in the following table:

System	Functionalities
Nokia	a) Problem: With mountains of data being generated every day, you need a solution to monitor your networks reliably, accurately and cost-efficiently. b)Tool: The new Capacity Advisor feature helps you predict the network capacity needs, and it recommends what actions to take and when. Thanks to this you can ensure that there is enough capacity in the network when you roll out your new service. c) They used the dashboards in KPI prediction and time slot classification. For showing as many dimensions as possible and showing relations between data. They used them in Predictive operations as the following: - Service Key Performance Indicator (KPI) monitor system health. - SLA agreement guarantees 99.x percent availability and steady values for normal operation.(1)
NSN	a) Monitor network alarm and create a Trouble Ticket if needed after initial checks are done. b) Log the exact time of the alarm, as it appeared in OSS. c) Sent SMS to management team if the case is considered as critical d) Notify the customer through emails for outage issues as per management directions. e) They used the dashboards in the User Mobility. -For showing Usage of traffic planning, ads planning and traffic jam prediction. -They also used them in cell classification and predicts preferred user movement.(2)
The Proposed Tool	a) It will collect the files with different formats from the network nodes. b) Then, parsing them and save them in the database. c) There are some operations that can be made on the parsed files as the following: - classification, clustering, and data analysis. - A website for displaying the dashboards, the results of the operations that is done before, and the whole capacity monitoring process. - A mobile application will be available with a custom dashboard for viewing the data summary and receiving alarms.

2.3 User Characteristics

The characteristics of the user are a good experience with the computer applications, have a perfect background about the network components. Also, the user must have a good experience with data analysis as clustering and classification. So the capacity team and administrator are responsible for monitoring network resources through the web-based graphical user interface through visualizing the dashboards and perform operations.

2.4 User Problem Statement

Vodafone has a crucial problem of monitoring the massive capacity of their network nodes in real time. The late response for such an issue might result in complete or partial shutdown of a major server or an active node.

2.5 User Objectives

The capacity team needs a tool that is able to collect different files with different formats from different nodes on regular basis (daily, hourly...etc.). This tool will parse the received files from the node and insert the needed data into the database. Also, a notification system is needed in case the capacity of a certain node is not enough either via sending an SMS or an e-mail and Figures and analysis of data will be shown on the dashboard website. There are some analysis operations that can be made on these files as the following average, sum, maximum and minimum to produce configured data.

2.6 General Constraints

The general constraint is connecting to Vodafone access server.

3 Functional Requirements

Name	Define algorithm attributes
Type	Function requirements
Criticality	High
Input	//
Output	Selected algorithm and its attributes
Description	define the algorithm used and its attributes.
Priority	High
Expected risks	
Preconditions	calling selected data from database
Post-conditions	applying the algorithm on the selected data.

Name	Define algorithm parameters
Type	Function requirements
Criticality	High
Input	//
Output	defined parameters
Description	define the parameters of the used algorithm, this is customized per user.
Priority	High
Expected risks	
Preconditions	calling selected data from database
Post-conditions	applying the algorithm on the selected data.

Name	classify
Type	Function requirements
Criticality	High
Input	selected Algorithm and selected dataset
Output	classified data
Description	classification for the collected data.
Priority	High
Expected risks	data error, logical error.
Preconditions	calling data from database
Post-conditions	store the classified data into database.

Name	analyze data
Type	Function requirements
Criticality	High
Input	selected Algorithm and selected data
Output	analyzed data
Description	data analysis for the selected data.
Priority	High
Expected risks	data error, missing design, logical error.
Preconditions	calling data from database
Post-conditions	store the classified data into database.

Name	clustering
Type	Function requirements
Criticality	High
Input	selected Algorithm and selected data
Output	clustered data
Description	dividing the selected data for groups using clusters.
Priority	High
Expected risks	gathering the data around one cluster and the rest nothing.
Preconditions	calling data from database
Post-conditions	store the clusters data into database.

Name	getDataSet
Type	Function requirements
Criticality	Medium
Input	Parameters for specific data set
Output	The Required data set
Description	Getting data set from the database to make the required operation.
Priority	High
Expected risks	dropping the connection between the database.
Preconditions	calling database
Post-conditions	sending data set to the capacity management team or admin

Name	declare certain capacity
Type	Function requirements
Criticality	high
Input	The required node
Output	The required capacity for the node.
Description	Declaring the capacity of each node by each admin.
Priority	High
Expected risks	
Preconditions	define the value of capacity
Post-conditions	set the value of the capacity

Name	receive file
Type	Function requirements
Criticality	Medium
Input	The data set and the path of the required file.
Output	The required file.
Description	This is for getting the file from the database.
Priority	Medium
Expected risks	Error while receiving the file or receiving the wrong file.
Preconditions	collect the file
Post-conditions	store it into the database

Name	Display dashboards
Type	Function requirements
Criticality	High
Input	The parameters for the details of the specific dashboard.
Output	Displaying the dashboard.
Description	Display the dashboard for making the visualization of the data clearly.
Priority	High
Expected risks	
Preconditions	collecting the required details
Post-conditions	sending the dashboards to the capacity management team or admin

Name	Receive alarm
Type	Function requirements
Criticality	High
Input	//
Output	Generating alarm
Description	when the node reach its defined capacity so it sends an alarm.
Priority	High
Expected risks	
Preconditions	knowing the value of the capacity of the node.
Post-conditions	receiving the alarm to one of the capacity management team.

Name	Display alarm
Type	Function requirements
Criticality	High
Input	//
Output	display alarm
Description	showing the display of the alarm.
Priority	High
Expected risks	
Preconditions	checking on the current capacity of the node.
Post-conditions	sending alarm for the capacity team.

Name	Update user data
Type	Function requirements
Criticality	High
Input	The old data
Output	The updated data
Description	when the admin wants to update the data.
Priority	High
Expected risks	
Preconditions	getting the required data to be updated.
Post-conditions	replacing the old data by the updates data.

Name	Display report
Type	Function requirements
Criticality	high
Input	the required details
Output	Displaying the report
Description	for generating the reports and visulize it.
Priority	High
Expected risks	
Preconditions	gathering all the details from the database.
Post-conditions	displaying the report for the capacity management team or admin.

Name	Search data
Type	Function requirements
Criticality	Medium
Input	Data set
Output	required data
Description	searching for the required data in the database.
Priority	High
Expected risks	unlinked database or syntax error
Preconditions	setting the database
Post-conditions	display the searched data for the asked user.

Name	display charts
Type	Function requirements
Criticality	High
Input	required details for the display charts
Output	displaying charts
Description	displaying the data with charts
Priority	High
Expected risks	
Preconditions	collecting the required data for displaying the charts.
Post-conditions	displaying the charts for the capacity management team and admin.

Name	Sign up
Type	Function requirements
Criticality	Medium
Input	user data details
Output	storing the data in the database
Description	registering the new user
Priority	High
Expected risks	
Preconditions	calling database
Post-conditions	storing the user data into the database.

Name	Log in
Type	Function requirements
Criticality	Medium
Input	user details (Email and password)
Output	display the interface for the user.
Description	it is a combination of information that authenticates your identity
Priority	High
Expected risks	
Preconditions	calling database
Post-conditions	log in the user

Name	Select data
Type	Function requirements
Criticality	medium
Input	selected data
Output	display selected data
Description	selecting certain data from the database.
Priority	High
Expected risks	
Preconditions	calling database
Post-conditions	display the selected data for capacity management team and admin.

Name	Choose operation
Type	Function requirements
Criticality	High
Input	operation type and data
Output	result for the operation
Description	this is for choosing the operation to be done on the selected data.
Priority	High
Expected risks	
Preconditions	calling the database
Post-conditions	the result of the operation done on the selected data.

Name	add user
Type	Function requirements
Criticality	high
Input	user details
Output	store user details
Description	letting the admin to add a user.
Priority	High
Expected risks	
Preconditions	calling the database
Post-conditions	giving permission to add a user

Name	Delete user
Type	Function requirements
Criticality	medium
Input	selected user
Output	display selected user
Description	Deleting the selected user from the database.
Priority	High
Expected risks	
Preconditions	calling database
Post-conditions	Deleting the user from the database.

Name	get node current capacity
Type	Function requirements
Criticality	high
Input	node details
Output	node capacity
Description	getting the node capacity for checking the capacity.
Priority	High
Expected risks	
Preconditions	calling the node details from database
Post-conditions	sending the node capacity to the capacity management team or admin.

Name	send alarm
Type	Function requirements
Criticality	high
Input	//
Output	sending alarm
Description	sending alarms to the admin or the capacity management team.
Priority	High
Expected risks	
Preconditions	check if the capacity of the node is above the target.
Post-conditions	sending the alarm to the admin or the capacity management team.

Name	Store data of user
Type	Function requirements
Criticality	medium
Input	user details
Output	//
Description	storing the user details in the database.
Priority	High
Expected risks	
Preconditions	calling database
Post-conditions	storing the user details in the database.

Name	generate report
Type	Function requirements
Criticality	high
Input	required data for makong report
Output	report
Description	gathering the data for making a report on the nodes.
Priority	High
Expected risks	
Preconditions	calling database
Post-conditions	display the report for capacity management team and admin.

Name	store parsed data
Type	Function requirements
Criticality	high
Input	data sets
Output	stored parsed data
Description	parsing the collected files from the node.
Priority	High
Expected risks	
Preconditions	collecting files and calling database
Post-conditions	storing the parsed files into the database.

Name	Select data
Type	Function requirements
Criticality	medium
Input	selected data
Output	display selected data
Description	selecting certain data from the database.
Priority	High
Expected risks	
Preconditions	calling database
Post-conditions	display the selected data for capacity management team and admin.

Name	parse TXT
Type	Function requirements
Criticality	high
Input	collected text files
Output	parsed file
Description	parsing the collected text files from the nodes.
Priority	High
Expected risks	collect a different format
Preconditions	collecting files
Post-conditions	store the parsed files into the database.

Name	parse CSV
Type	Function requirements
Criticality	high
Input	collected CSV files
Output	parsed file
Description	parsing the collected CSV files from the nodes.
Priority	High
Expected risks	collect a different format
Preconditions	collecting files
Post-conditions	store the parsed files into the database.

Name	parse XML
Type	Function requirements
Criticality	high
Input	collected XML files
Output	parsed file
Description	parsing the collected XML files from the nodes.
Priority	High
Expected risks	collect a different format
Preconditions	collecting files
Post-conditions	store the parsed files into the database.

Name	parse STAT
Type	Function requirements
Criticality	high
Input	collected STAT files
Output	parsed file
Description	parsing the collected STAT files from the nodes.
Priority	High
Expected risks	collect a different format
Preconditions	collecting files
Post-conditions	store the parsed files into the database.

Name	parse LOG
Type	Function requirements
Criticality	high
Input	collected LOG files
Output	parsed file
Description	parsing the collected LOG files from the nodes.
Priority	High
Expected risks	collect a different format
Preconditions	collecting files
Post-conditions	store the parsed files into the database.

4 Interface Requirements

This section describes how Vodafone worker will interact with the system.

4.1 Web Based Graphical User Interface

The system website works on showing data in charts, display/make operations like ”cluster”, ”classify”, ”parse”, ”data analysis” on the data. The mobile application works on showing data and its summary, notify with the updated report.

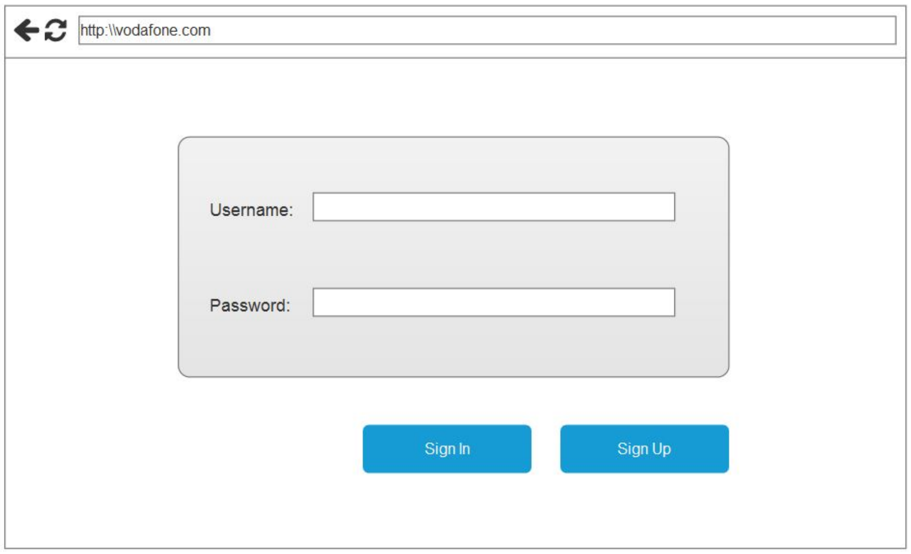


Figure 3: Sign-In Page

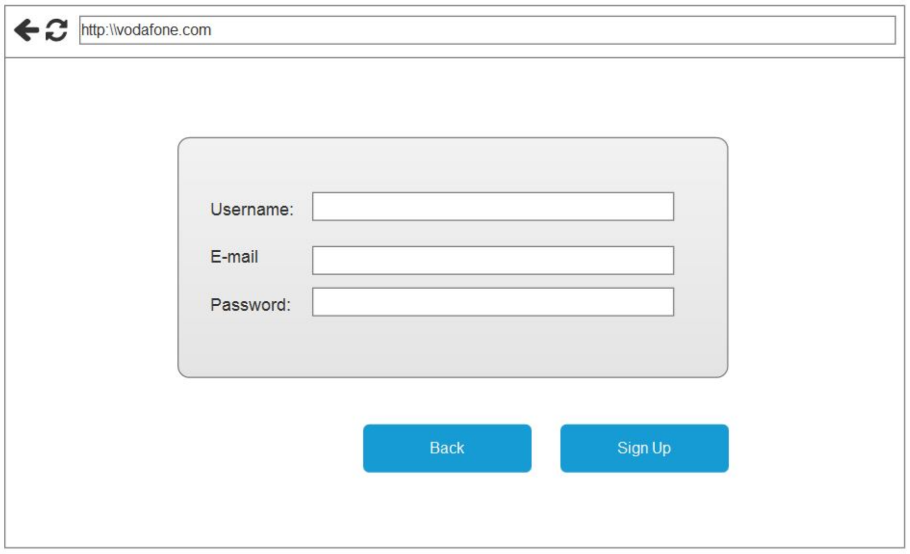
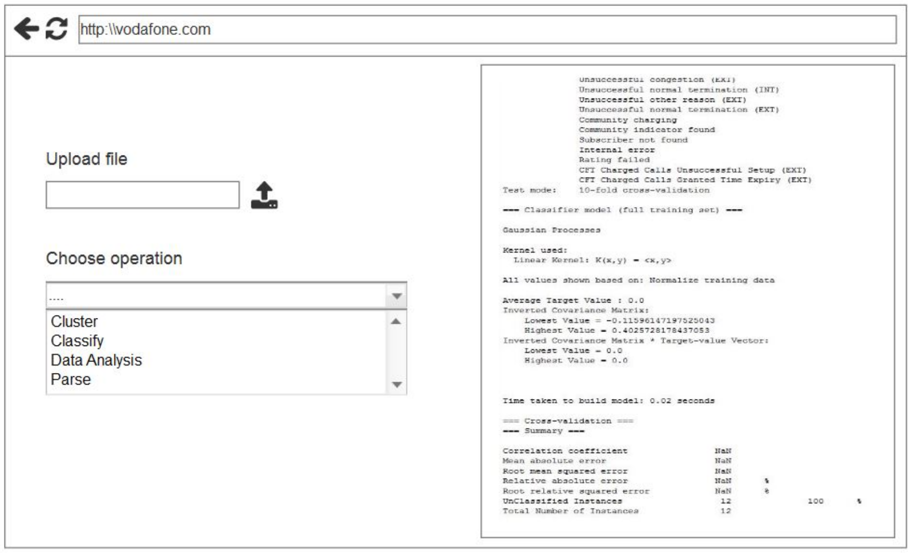


Figure 4: Sign-Up Page



```
              Unsuccessful congestion (EXT)
              Unsuccessful normal termination (INT)
              Unsuccessful other reason (EXT)
              Unsuccessful normal termination (EXT)
              Community charging
              Community indicator found
              Subscriber not found
              Internal error
              Rating failed
              CFT Charged Calls Unsuccessful Setup (EXT)
              CFT Charged Calls Granted Time Expiry (EXT)
Test mode:    10-fold cross-validation

--- Classifier model (full training set) ---

Gaussian Processes

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.0
Inverted Covariance Matrix:
    Lowest Value = -0.11596147197525043
    Highest Value = 0.4025728178437053
Inverted Covariance Matrix * Target-value Vector:
    Lowest Value = 0.0
    Highest Value = 0.0


Time taken to build model: 0.02 seconds

=== Cross-validation ===
--- Summary ---

Correlation coefficient                  NaN
Mean absolute error                      NaN
Root mean squared error                  NaN
Relative absolute error                  NaN      %
Root relative squared error              NaN      %
UnClassified Instances                    12              100      %
Total Number of Instances                 12
```

Figure 5: Upload files with different formats and choose an operation such as parsing, classifying, clustering, or data analysing

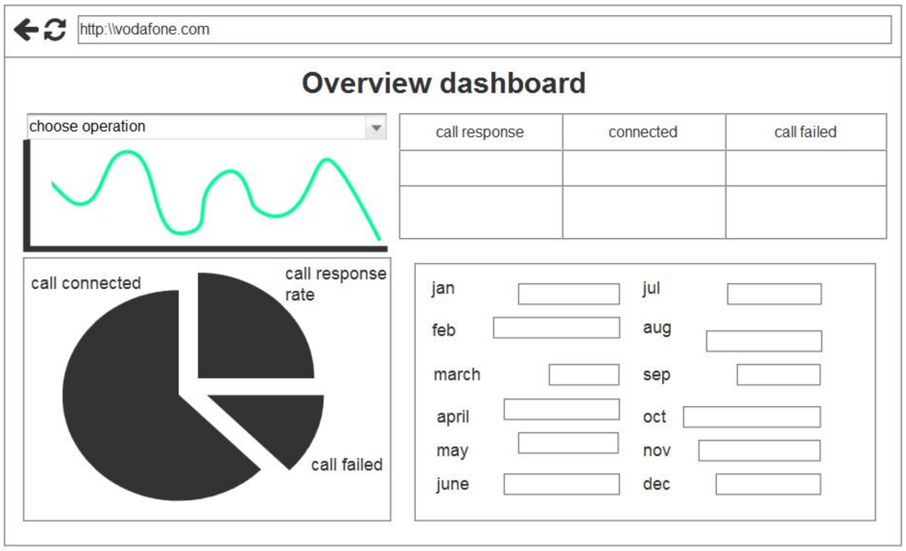


Figure 6: Overview for data in dashboards (charts)

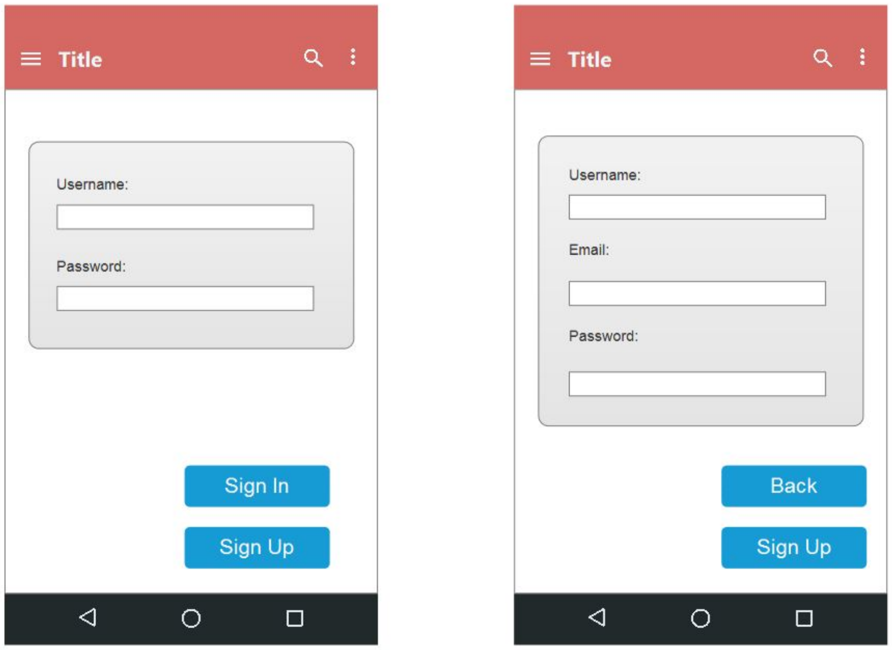
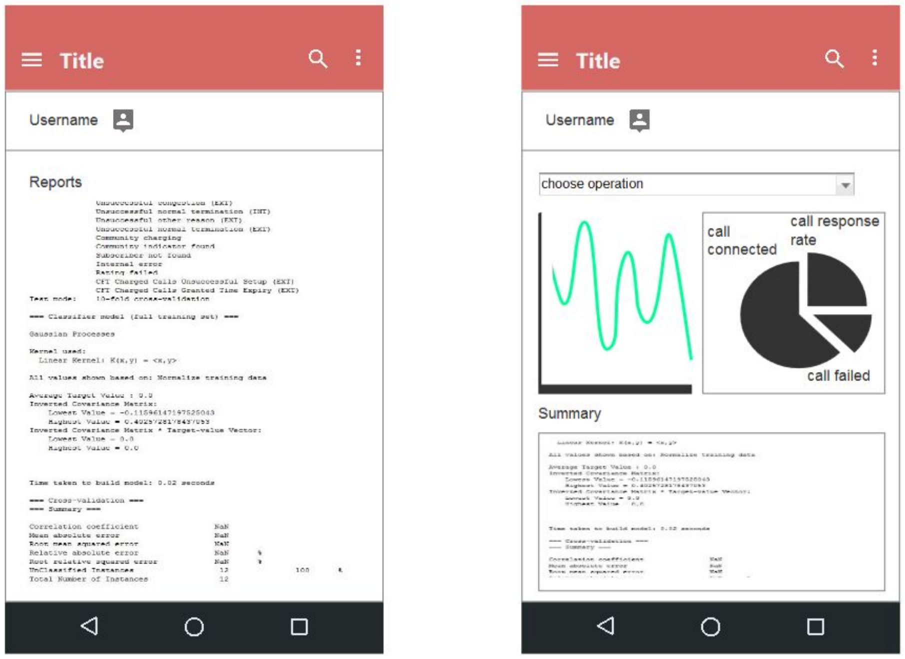


Figure 7: Sign-in and sign-up pages for mobile application



```
              Unsuccessful congestion (EXT)
              Unsuccessful normal termination (INT)
              Unsuccessful other reason (EXT)
              Unsuccessful normal termination (EXT)
              Community charging
              Community indicator found
              Subscriber not found
              Internal error
              Rating failed
              CFT Charged Calls Unsuccessful Setup (EXT)
              CFT Charged Calls Granted Time Expiry (EXT)
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Gaussian Processes

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.0
Inverted Covariance Matrix:
    Lowest Value = -0.11596147197525043
    Highest Value = 0.40257281784370 53
Inverted Covariance Matrix * Target-value Vector:
    Lowest Value = 0.0
    Highest Value = 0.0



Time taken to build model: 0.02 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                 NaN
Mean absolute error                     NaN
Root mean squared error                 NaN
Relative absolute error                 NaN      %
Root relative squared error             NaN      %
UnClassified Instances                   12               100      %
Total Number of Instances                12
```

```
  Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.0
Inverted Covariance Matrix:
    Lowest Value = -0.11596147197525043
    Highest Value = 0.40257281784370 53
Inverted Covariance Matrix * Target-value Vector:
    Lowest Value = 0.0
    Highest Value = 0.0



Time taken to build model: 0.02 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                 NaN
Mean absolute error                     NaN
Root mean squared error                 NaN
```

Figure 8: Reports sent daily, hourly.. from the system and displaying data in charts with summary

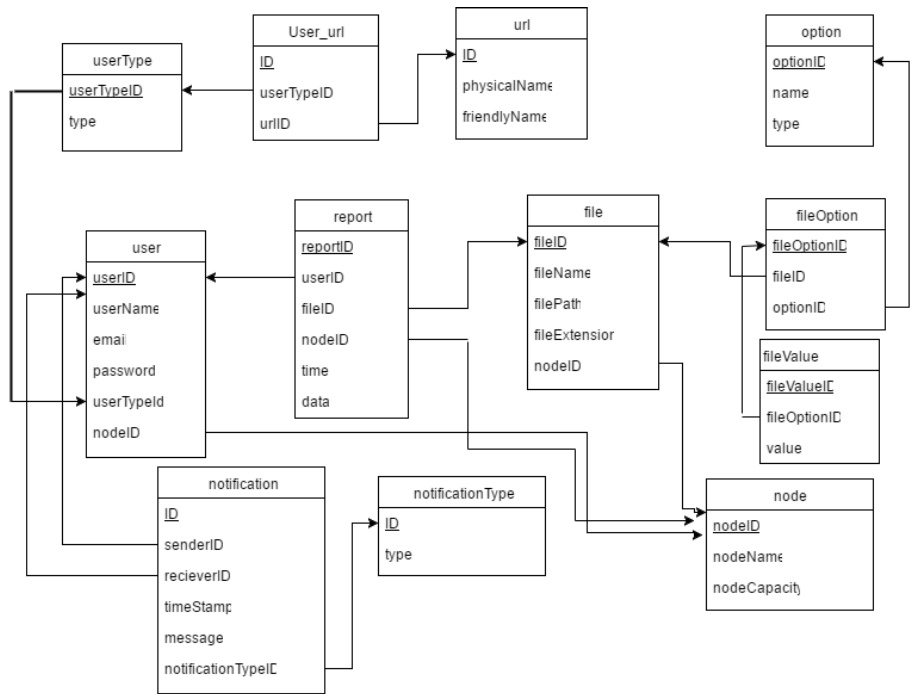


Figure 9: Database

5 Other non-functional attributes

5.1 Security

All files will be access-able only by Vodafone server.

5.2 Portability

A mobile application will be made for receiving alarms in case of the capacity of a certain node is not enough either via sending an SMS or an e-mail. Also, the mobile application will display charts data, and a website for displaying the dashboards of the nodes viewed for the capacity team on the parsed files.

6 Database

We designed these tables to be compatible with the data set and requirements of Vodafone company.

6.1 File, File Option , File Value and Option

In these four tables, we used the entity attribute value design to be compatible with Vodafone data sets samples. We used the entity attribute value because the structure of the data sets is ambiguous and unclear and to avoid the null values.

6.2 User, User Type , User URL and URL

These tables designed for storing user information (administrator and capacity management team) as username, password and email. 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.

6.3 Node

This designed table is for storing the nodes' name and its capacity.

6.4 Report

The objective of this table is for generating reports for the end user. The report specifies the nodes' details, users' details, files' details , current time and date.

6.5 Notification and Notification Type

These tables are designed for notify the users with a SMS or an E-mail notification.

7 Preliminary Object-Oriented Domain Analysis

This section presents a list of the fundamental objects that must be modeled within the system to satisfy its requirements. A primitive class diagram is as follows:

7.1 Classes Descriptions

7.1.1 connectionDB

This class is a concrete class.

7.1.2 List of Superclasses:

No super classes for this class.

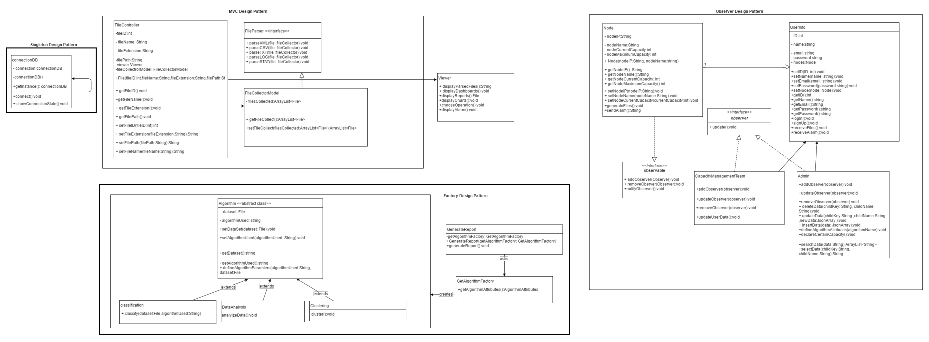

Singleton Design Pattern
connectionDB
- connection:connectionDB
-connectionDB()
+getInstance(): connectionDB
+connect():void
+ showConnectionState():void
MVC Design Pattern
FileController
-fileID:int
- fileName: String
- fileExtension:String
-filePath:String
-viewer:Viewer
-fileCollectrorModel: FileCollectorModel
+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>>
+ parseXML(file :fileCollector) void
+ parseCSV(file :fileCollector) void
+ parseTXT(file :fileCollector) void
+ parseLOG(file :fileCollector) void
+ parseSTAT(file :fileCollector) 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
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
+ removeOberver(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
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
extends
classification
+ classify(dataset:File,algorithmUsed:String)
DataAnalysis
analyizeData():void
Clustering
cluster():void
GenerateReport
-getAlgorithmFactory: GetAlgorithmFactory
+GenerateReport(getAlgorithmFactory: GetAlgorithmFactory)
+generateReport():void
asks
GetAlgorithmFactory
+getAlgorithmAttributes():AlgorithmAttributes
creates

Figure 10: Primitive Class Diagram

7.1.3 List of Subclasses:

No sub-classes for this class.

7.1.4 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.

7.1.5 Collaborations:

No collaborations are needed.

7.1.6 Attributes:

connection: is an instance of the connectionDB class.

7.1.7 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.

7.1.8 Constraints:

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

7.1.9 FileCollectorModel

This class is a concrete class.

7.1.10 List of Superclasses:

No super classes for this class.

7.1.11 List of Subclasses:

No sub-classes for this class.

7.1.12 Purpose:

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

7.1.13 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.

7.1.14 Attributes:

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

7.1.15 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.

7.1.16 Constraints:

No constraints.

7.1.17 FileController

This class is a concrete class.

7.1.18 List of Superclasses:

No super classes for this class.

7.1.19 List of Subclasses:

No sub-classes for this class.

7.1.20 Purpose:

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

7.1.21 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.

7.1.22 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.

7.1.23 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.

7.1.24 Constraints:

No constraints.

7.1.25 FileParser

This is an interface.

7.1.26 List of Superclasses:

No super classes for this class.

7.1.27 List of Subclasses:

No sub-classes for this class.

7.1.28 Purpose:

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

7.1.29 Collaborations:

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

7.1.30 Attributes:

No attributes.

7.1.31 Operations

: parseXML(file :fileCollector): has no return value. Its purpose is to parse files with extensible markuo language(xml) format. Its implementation will be in the FileCollectorModel class that implements this interface.

parseCSV(file :fileCollector): has no return value. Its purpose is to parse files with comma seperated value(csv) format. Its implementation will be in the FileCollectorModel class that implements this interface.

parseTXT(file :fileCollector) : has no return value. Its purpose is to parse files with text format. Its implementation will be in the FileCollectorModel class that implements this interface.

parseLOG(file :fileCollector) : has no return value. Its purpose is to parse files with log format. Its implementation will be in the FileCollectorModel class that implements this interface.

parseSTAT(file :fileCollector): has no return value. Its purpose is to parse files with stat format. Its implementation will be in the FileCollectorModel class that implements this interface.

7.1.32 Constraints:

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

7.1.33 Viewer

This class is a concrete class.

7.1.34 List of Superclasses:

No super classes for this class.

7.1.35 List of Subclasses:

No sub-classes for this class.

7.1.36 Purpose:

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

7.1.37 Collaborations:

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

7.1.38 Attributes:

No attributes.

7.1.39 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.

7.1.40 Constraints:

No constraints.

7.1.41 Node

This class is a concrete class.

7.1.42 List of Superclasses:

No super classes for this class.

7.1.43 List of Subclasses:

No sub-classes for this class.

7.1.44 Purpose:

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

7.1.45 Collaborations:

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

7.1.46 Attributes:

nodeIP, and nodeName: are of type String.

nodeCurrentCapacity, and nodeMaximumCapacity: are of type integer.

7.1.47 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.

7.1.48 Constraints:

No constraints.

7.1.49 Admin

This class is a concrete class.

7.1.50 List of Superclasses:

It extends from super-class ”UserInfo”.

7.1.51 List of Subclasses:

No sub-classes for this class.

7.1.52 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.

7.1.53 Collaborations:

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

7.1.54 Attributes:

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

7.1.55 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.

7.1.56 Constraints:

No constraints.

7.1.57 CapacityManagementTeam

This class is a concrete class.

7.1.58 List of Superclasses:

It extends from super-class ”UserInfo”.

7.1.59 List of Subclasses:

No sub-classes for this class.

7.1.60 Purpose:

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

7.1.61 Collaborations:

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

7.1.62 Attributes:

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

7.1.63 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.

7.1.64 Constraints:

No constraints.

7.2 Classification , Clustering and DataAnalysis

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

7.2.1 List of Superclasses:

No super classes for this class.

7.2.2 Purpose:

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

7.2.3 Collaborations:

No collaborations are needed. Attributes:

7.2.4 Operations

: classify(dataset: File, algorithmUsed:String) its the function that classify data in files.

analyzeData(): has no return value.

cluster(): has no return value, Its purpose is to cluster the data.

7.2.5 Constraints:

No constraints.

7.3 Algorithm

This class is an abstract class.

7.3.1 List of Superclasses:

No super classes for this class.

7.3.2 List of Subclasses:

subclasses: classification, dataAnalysis, Clustering

7.3.3 Purpose:

The basic purpose of this class is to choose an algorithm to classify, or cluster, or analys data.

7.3.4 Collaborations:

No collaborations are needed. Attributes:

7.3.5 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.

7.3.6 Constraints:

No constraints.

7.4 Classification , Clustering , DataAnalysis

These classes are supclasses extends from Class "Algorithm".

7.4.1 List of Superclasses:

No super classes for this class.

7.4.2 Purpose:

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

7.4.3 Collaborations:

No collaborations are needed. Attributes:

7.4.4 Ope rations

: classify(dataset: File, algorithmUsed:String): has no return value. Its purpose is to classify data.

analyzeData(): has no return value. Its purpose is to analyze data.

cluster(): has no return value, Its purpose is to cluster data.

7.4.5 Constraints:

7.5 UserInfo

This class is a super-class

7.5.1 Subclasses:

Sub-classes are : Admin and CapacityManegmentTeam

7.5.2 Purpose:

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

7.5.3 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.

7.5.4 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.

7.5.5 Constraints:

7.6 observable

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

7.6.1 Subclasses:

No subclasses.

7.6.2 Purpose:

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

7.6.3 Collaborations:

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

7.7 observer

This is an interface for updating the notifications.

7.7.1 Subclasses:

No subclasses.

7.7.2 Purpose:

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

7.7.3 Collaborations:

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

7.7.4 List of Superclasses:

Names all immediate superclasses.

7.7.5 List of Subclasses:

Names all immediate subclasses.

7.7.6 Purpose:

States the basic purpose of the class.

7.7.7 Collaborations:

Names each class with which this class must interact in order to accomplish its purpose, and how.

7.7.8 Attributes:

Lists each attribute (state variable) associated with each instance of this class, and indicates examples of possible values (or a range).

7.7.9 Operations

: Lists each operation that can be invoked upon instances of this class. For each operation, the arguments (and their type), the return value (and its type), and any side effects of the operation should be specified.

7.7.10 Constraints:

Lists any restrictions upon the general state or behavior of instances of this class.

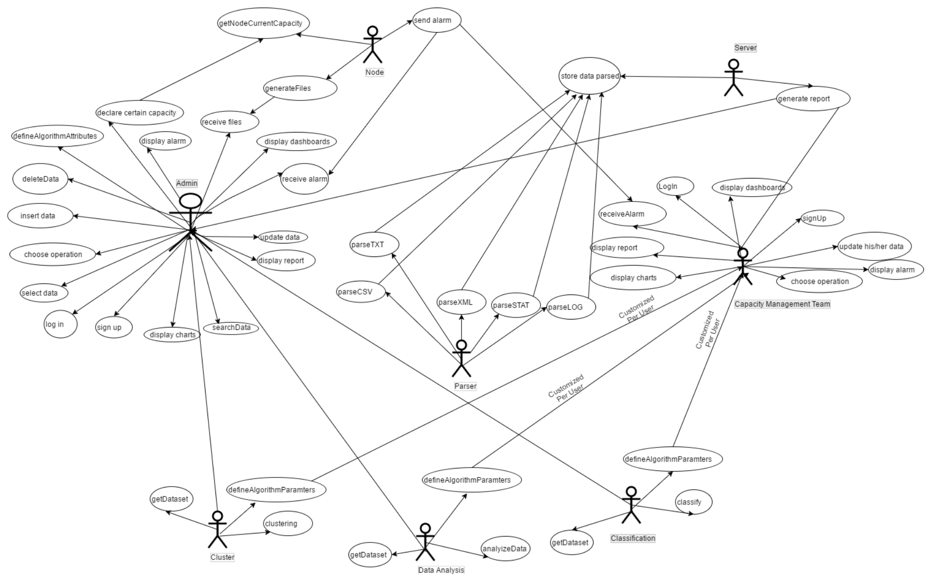


Figure 11: Vodafone UseCase

8 Operational Scenarios

1)Declare certain capacity :

The admin could declare certain capacity for each node.

2)Display alarm :

The admin could display the alarm when the capacity of a node reaches more than the declared capacity.

3)receive files :

The admin receives all the files which are generated from all the nodes.

4)Display dashboards :

The admin could display dashboards

5)receive alarms :

The admin will receive alarm information like the name of the node and the capacity node that reached.

6)Update data :

The admin could update any data in the system.

7)Insert data :

The admin could insert any kind of data in the system.

8)Delete data :

The admin could delete any kind of data in the system.

9)Select data :

The admin could select any data he/she wanted from the system.

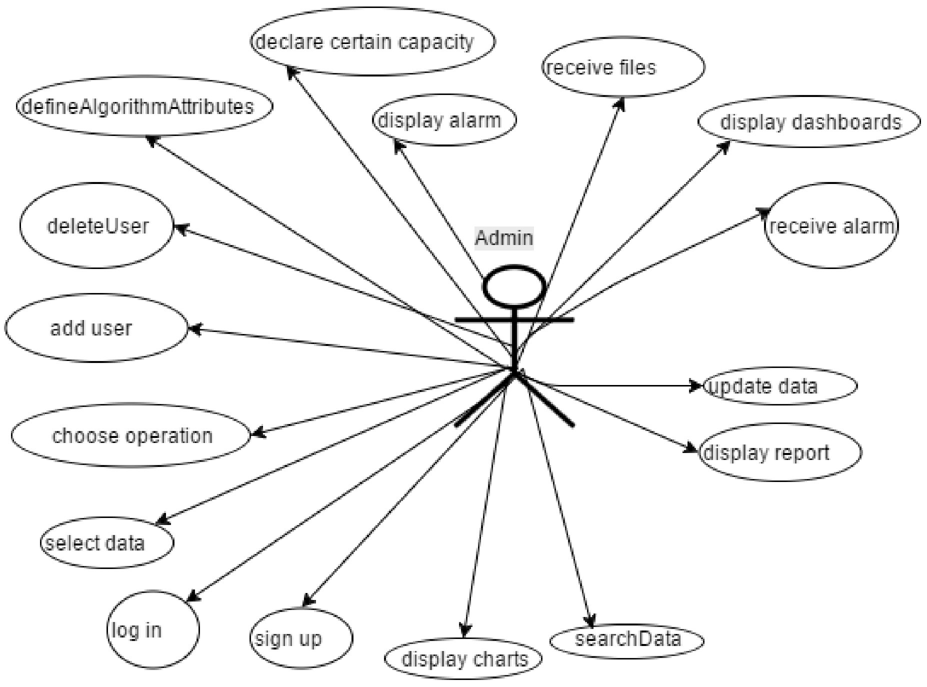


Figure 12: Admin UseCase

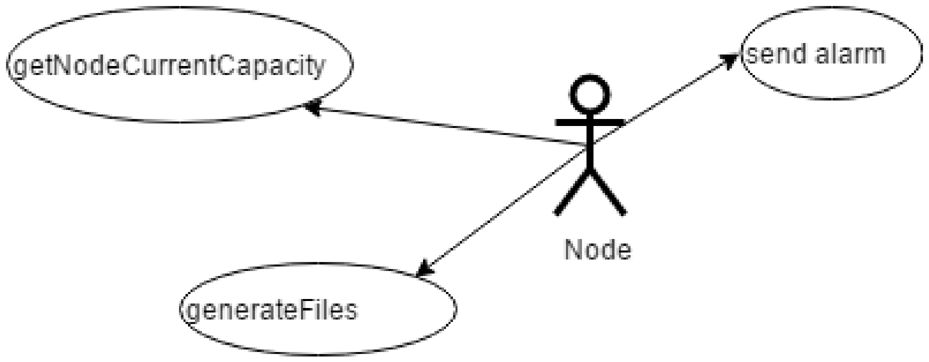


Figure 13: Node UseCase

10)Search data :
The admin could search what ever he/she wanted to find from the system.
11)Display report :
The admin could display the generated report.
12)Display charts :
The admin could display the generated charts.
13)Sign up :
The admin must sign up and enter his data like username, email, and password.
14)Log in:
The admin must sign in using email and password to enter to the admin pages.
15)Choose Operation :
The admin could choose the operation to implement on the parsed data.
16)Define Algorithm Attributes:
The admin would declare the algorithm attributes and declare default values according to the parameters of the algorithm.
1)Send alarm :
The node will send an alarm to the admin and the capacity management team if the node reached more than the declared capacity.
2)Generate files :
Each node will generate files with different extensions.
3)Get node current capacity
The node will be declared its capacity.
1)Generate report :
The server generates all the reports.
2)Store data parsed :
The server is the place where the data which is parsed would be stored in it.

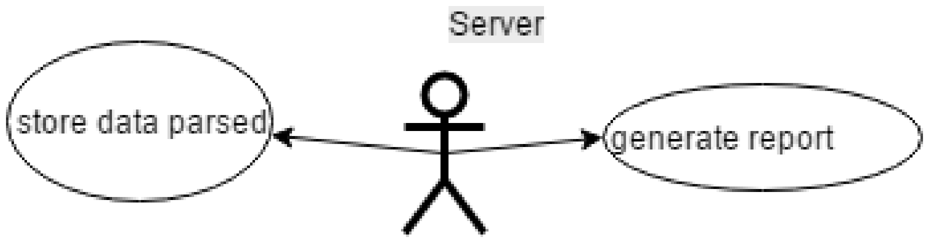


Figure 14: Server UseCase

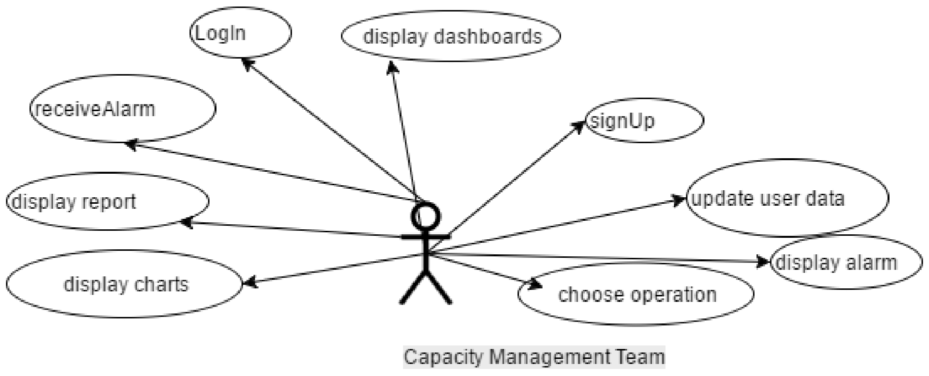


Figure 15: Capacity Monitoring UseCase

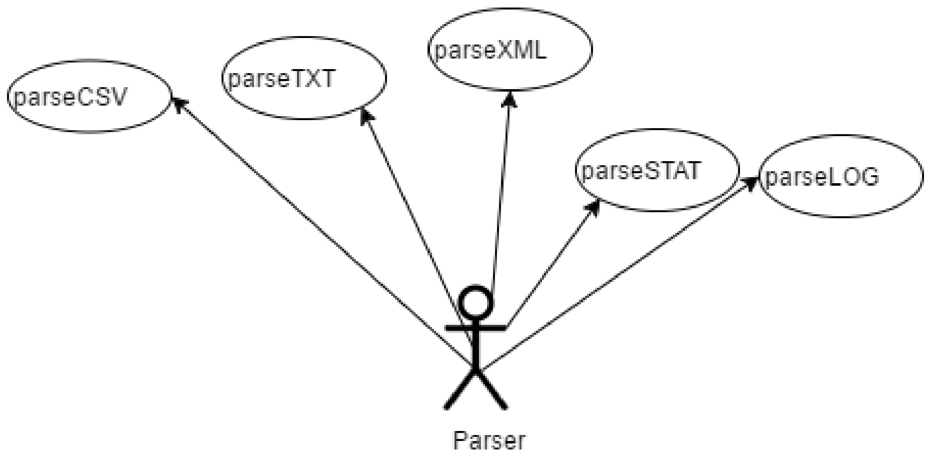


Figure 16: Parser UseCase

1) Sign Up :

All of the capacity management team must sign up writing their username, email, and password.

2)Update User Data

The member of the capacity management team could update their own information like username, email, and password.

3)Display Alarm :

The member of capacity management team could display the received alarm.

4)Choose Operation :

The member of capacity management team could choose an algorithm to implemented it on the parsed data.

5)Log In:

The member of capacity management team must log in with their email and password to reach their own web pages.

6)Receive Alarm

The member of capacity management team receives alarm from the node that its capacity reached more than declared capacity.

7) Display Reports:

The member of capacity management team could display the reports which are generated.

8)Display Charts

The member of capacity management team could display the charts which are generated.

1)Parse TXT

This function is parsing the generated TXT files.

2)Parse CSV

This function is parsing the generated CSV files.

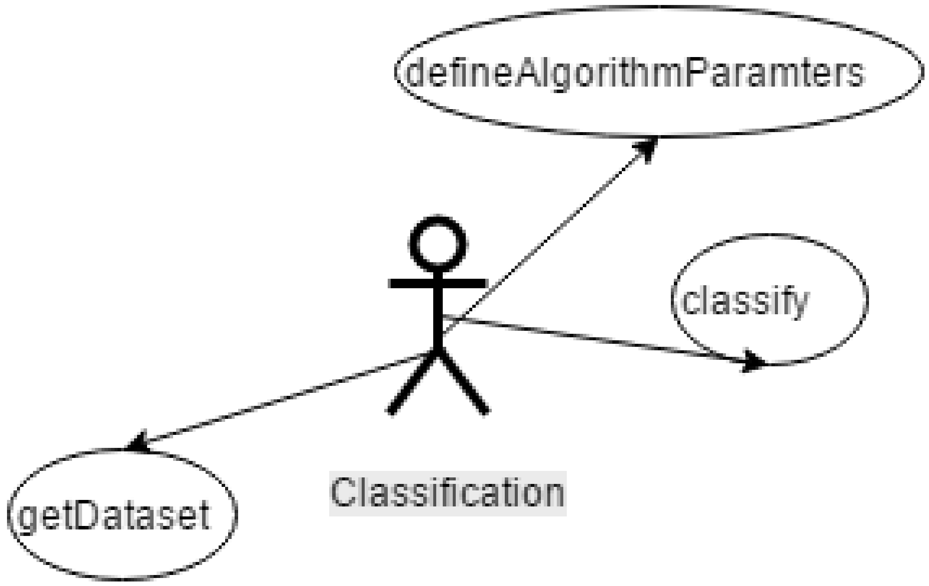


Figure 17: Classification UseCase

3)Parse XML

This function is parsing the generated XML files.

4)Parse STAT

This function is parsing the generated STAT files.

5)Parse LOG

This function is parsing the generated LOG files.

1)Get Dataset

The classification algorithms must have a dataset to be implemented.

2)Define Algorithm Parameters:

The classification algorithms must define its parameters.

3)Classify

This function is to implement the algorithms of the classification.

1)Get Dataset

The cluster algorithms must have a dataset to be implemented.

2)Define Algorithm Parameters:

The cluster algorithms must define its parameters.

3)Clustering

This function is to implement the algorithms of the clustering.

1)Get Dataset

The data analysis algorithms must have a dataset to be implemented.

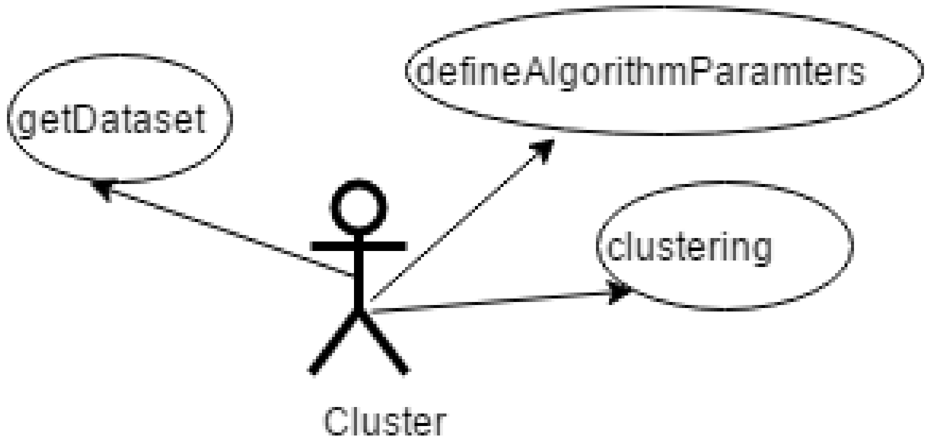


Figure 18: Cluster UseCase

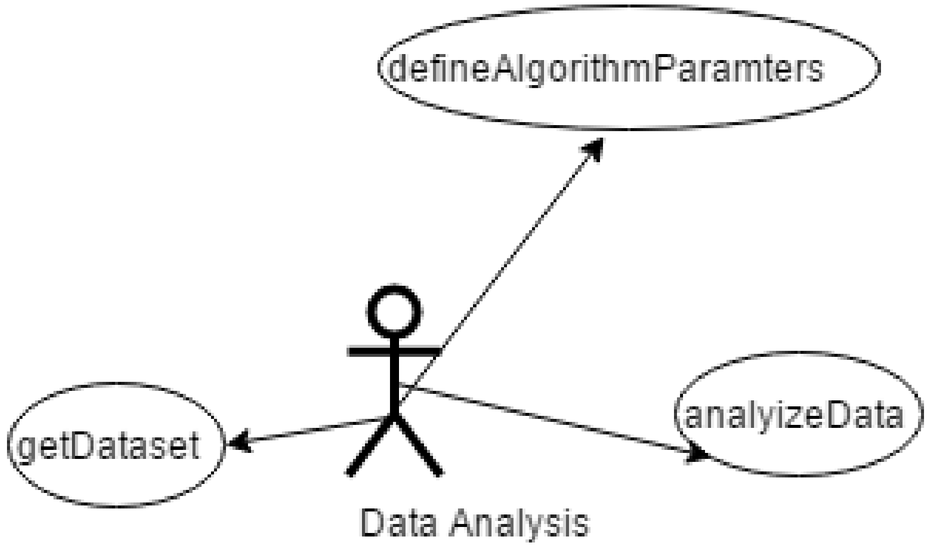
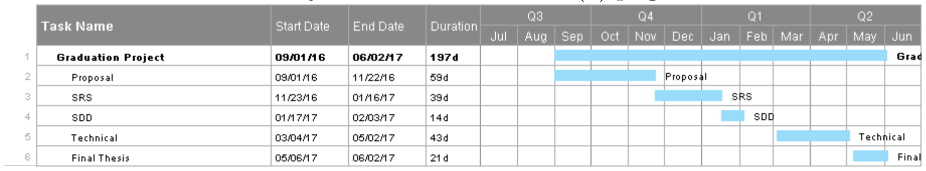


Figure 19: Data Analysis UseCase

Project with Gantt Chart (2).png



	Task Name	Start Date	End Date	Duration
1	**Graduation Project**	**09/01/16**	**06/02/17**	**197d**
2	Proposal	09/01/16	11/22/16	59d
3	SRS	11/23/16	01/16/17	39d
4	SDD	01/17/17	02/03/17	14d
5	Technical	03/04/17	05/02/17	43d
6	Final Thesis	05/06/17	06/02/17	21d

Figure 20: Gantt Chart For Preliminary Schedule

2)Define Algorithm Parameters:

The data analysis algorithms must define its parameters.

3)Analyze Data :

This function is to implement the algorithms of the data analysis.

9 Preliminary Schedule Adjusted

10 Preliminary Budget Adjusted

Item	Item Cost
Firebase Database	10 dollars

11 Appendices

Specifies other useful information for understanding the requirements. All SRS documents should include at least the following two appendices:

11.1 Definitions, Acronyms, Abbreviations

MS	Mobile Subscriber
BTS	Base Transceiver Station
BSC	Base Station Controller
BSC	Base Station Controller
RNC	Radio Network Controller
MSS	Mobile Switching Server
VLR	Visitor Location Register
HLR	Home Location Register
SGSN	Support GPRS Service Node
GGSN	Gateway GPRS Service Node.
MSC	Mobile Switching Center
SDP	Session Description Protocol / Service Data Point / Signal Distribution Panel.
RBT	Ring Back Tone.
RBD	Receiver Buffer Description.

11.2 Collected material

12 References

References

[1] [Online]. Available: https://networks.nokia.com/solutions/performance-manager

[2] [Online]. Available: https://www.expatriates.com/cls/31181204.html