A REPORT

ON

ALARM MONITORING AND AUTOMATION

BY

Name:	ID No:
JARED DOMINIC FERNANDEZ	2017B3A30588P

AT

VODAFONE IDEA LIMITED, HYDERABAD

A Practice School-I station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(July, 2019)

A REPORT

ON

ALARM MONITORING AND AUTOMATION

(Final Report)

BY

Name:	ID No:	Discipline:
JARED DOMINIC FERNANDEZ	2017B3A30588P	M.Sc. (Hons.) Economics, B.E.(Hons.) Electrical & Electronics

Prepared in partial fulfilment of the

Practice School-I Course No.

BITS F221/BITS F231/BITS F241

AT

VODAFONE IDEA LIMITED, HYDERABAD

A Practice School-I station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(July, 2019)

ACKNOWLEDGEMENTS

I would like to thank Vodafone Idea Limited (AP&T Circle) for giving me the opportunity to work at India's No: 1 Telcom company. I express my sincere gratitude to Mr Bhuban Chandra Bhatt, Deputy General Manager for Network Operations and Mr Anand Pothuri, Assistant Manager for Network Planning and Strategy, my mentors for this project, for their constant support, guidance and valuable advice. I am also grateful to Dr. R. N. Ponnalagu – PS Faculty for making all the necessary arrangements over the course of my internship at Vodafone Idea Limited, Hyderabad. I would also like to thank Mani Bhaskar Ma'am from HR, Vodafone Idea Limited for constant support. Finally, I appreciate the Practice school division's efforts for providing me with an opportunity to experience work done in a premier company over the course of Practice School – 1.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN)

Practice School Division

Station: VODAFONE IDEA LIMITED **Centre:** HYDERABAD

Duration: **From:** 21st May 2019 **To:** 13th July 2019

Date of Submission: 12th July 2019

Title of the Project: ALARM MONITORING AND AUTOMATION

ID No:	**Name**:	**Discipline:**
2017B3A30588P	JARED DOMINIC FERNANDEZ	M.Sc. (Hons.) Economics, B.E.(Hons.) Electrical & Electronics

Mentors and experts	**Designation**
Bhuban Chandra Bhatt	Deputy General Manager – Network Operations
Anand Pothuri	Assistant Manager – Network Planning and Strategy

Name of the PS Faculty: RN Ponnalagu

Key Words: Alarm Monitoring, Automation, Telecom, Python, Computer Programming, SQL, Database Management, Transmissions, Communication, Oracle Database, Data Analysis

Project Area: Telecommunication, Computer Science, Data Analysis

Abstract:

Faults in a telecommunication network are reported to management centres in the form of alarms. An *alarm* is a message emitted by a network element, typically when a problem is encountered. Unfortunately, a network element has a very narrow view of the network, and can therefore only report the symptoms of the fault from its limited viewpoint. On the other hand, one fault can result in a number of different alarms from several network elements. To combat this Alarm Monitoring is required. This helps in combating further problems. Automation of Procedure related to this will help make the detection and rectification process more fast and accurate. There are 9 major types of Alarms spanning across 2G, 3G and 4G Technology. This Report deals with these alarms, it's monitoring and an attempt at automating certain monitoring procedures.

Signature of Student	Signature of PS Faculty
Date	Date

TABLE OF CONTENTS

INTRODUCTION

Vodafone Idea Limited is an Aditya Birla Group and Vodafone Group partnership. It is India's leading telecom service provider. The Company provides pan India Voice and Data services across 2G, 3G and 4G platform. With the large spectrum portfolio to support the growing demand for data and voice, the company is committed to deliver delightful customer experiences and contribute towards creating a truly 'Digital India' by enabling millions of citizens to connect and build a better tomorrow. The Company is developing infrastructure to introduce newer and smarter technologies, making both retail and enterprise customers future ready with innovative offerings, conveniently accessible through an ecosystem of digital channels as well as extensive on-ground presence. Its vision is to create world-class digital experiences to connect and inspire every Indian to build a better tomorrow. The Company is listed on the National Stock Exchange (NSE) and Bombay Stock Exchange (BSE) in India.

In an industrial telecom environment, an "alarm" can be generated for a wide variety of reasons. In fact, some so-called "alarms" really aren't anything to worry about. For many companies, the routine act of opening a door will generate a low-severity "status" alarm. When a fault or event occurs, a network component will often send a notification to the network operator using a protocol. An alarm is a persistent indication of a fault that clears only when the triggering condition has been resolved. A current list of problems occurring on the network component is often kept in the form of an active alarm. Fault management systems may use complex filtering systems to assign alarms to severity levels.

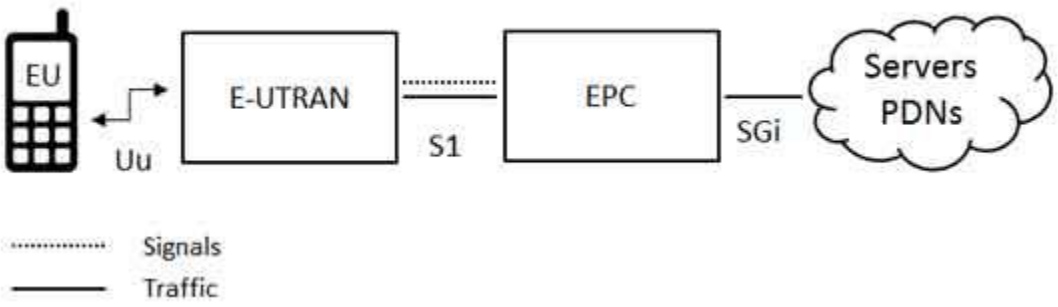
There are broadly nine different types of alarms in the Vodafone-Idea network. Of these five are related to 4G Technology, three are related to 3G Technology, and one pertains to 2G Technology. Alarm Monitoring is an integral part of ensuring sustained maintenance and improvement in the quality of the network. It is required that daily data is to be extracted from the Database Using SQL, further organised and then be mailed to the concerned party for the further continuation of Protocol. As part of my work here I needed to monitor Alarm data daily from the Oracle Database. To accomplish this I needed to acquire knowledge about the Alarm system and different components of Cellular Communication and it's Architecture. I have attempted to automate certain procedures onto a Graphical User Interface. I also created an elementary Web Page that can showcase the location of eNodeB's where there is a Live Alarm.

4G ARCHITECTURE

The high-level network architecture of LTE is comprised of the following three main components:

- The User Equipment (UE).
- The Evolved UMTS Terrestrial Radio Access Network (E-UTRAN).
- The Evolved Packet Core (EPC).

The evolved packet core communicates with packet data networks in the outside world such as the internet, private corporate networks or the IP multimedia subsystem. The interfaces between the different parts of the system are denoted Uu, S1 and SGi as shown below:



The E-UTRAN handles the radio communications between the mobile and the evolved packet core and just has one component, the evolved base stations, called **eNodeB** or **eNB**. Each eBN connects with the EPC by means of the S1 interface and it can also be connected to nearby base stations by the X2 interface, which is mainly used for signalling and packet forwarding during handover. Understanding the S1 interface is Integral to grasp how certain errors work.

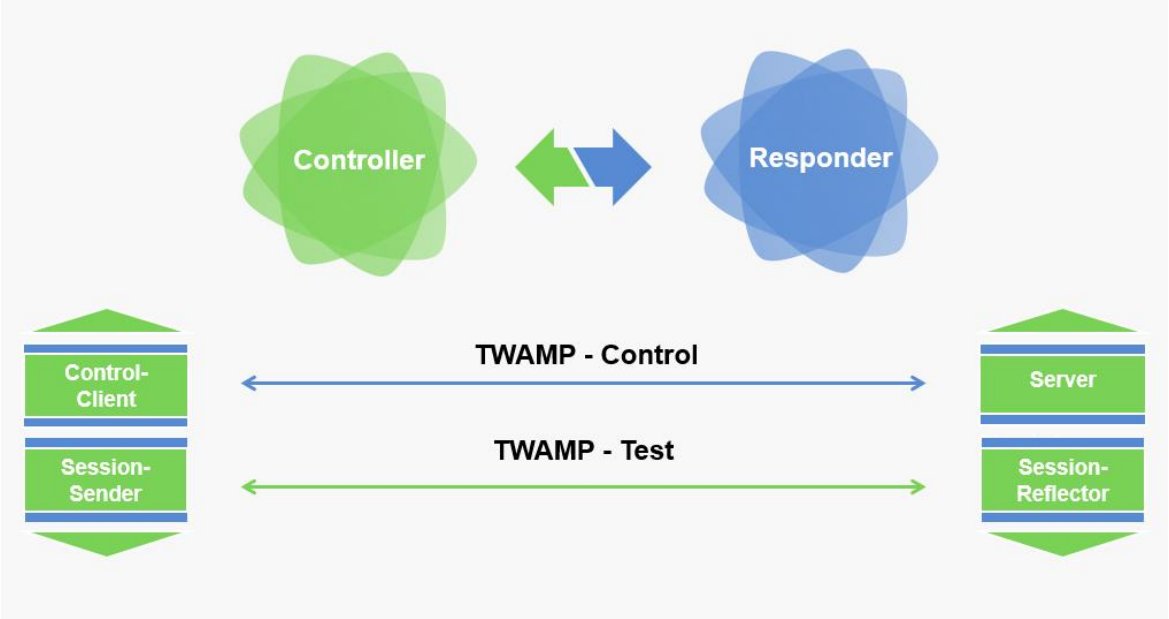
4G TECHNOLOGY ALARMS

TWAMP ALARM

The IETF Two-Way Active Measurement Protocol (TWAMP) defines a standard for measuring round-trip network performance between any two devices that support the TWAMP protocols. It accomplished this by sending dummy packets over the network measuring parameters like latency. The TWAMP architecture is composed of the following four logical entities:

- The control-client sets up, starts, and stops TWAMP-Test sessions.
- The session-sender instantiates TWAMP-Test packets that are sent to the session-reflector.
- The session-reflector reflects a measurement packet upon receiving a TWAMP-Test packet. It does not collect packet statistics in TWAMP.
- The TWAMP server is an end system that manages one or more TWAMP sessions.

Critical values arising from the TWAMP measurements give rise to the TWAMP Alarm



The figure above shows the four entities that make up the TWAMP architecture.

SYNCHRONIZATION ALARM

These alarms occur due to various synchronization problems that may occur at different interfaces in the network. Synchronization between a receiver and a sender is essential to ensure flaw-free communication.

1. SyncE SSM Timed Out

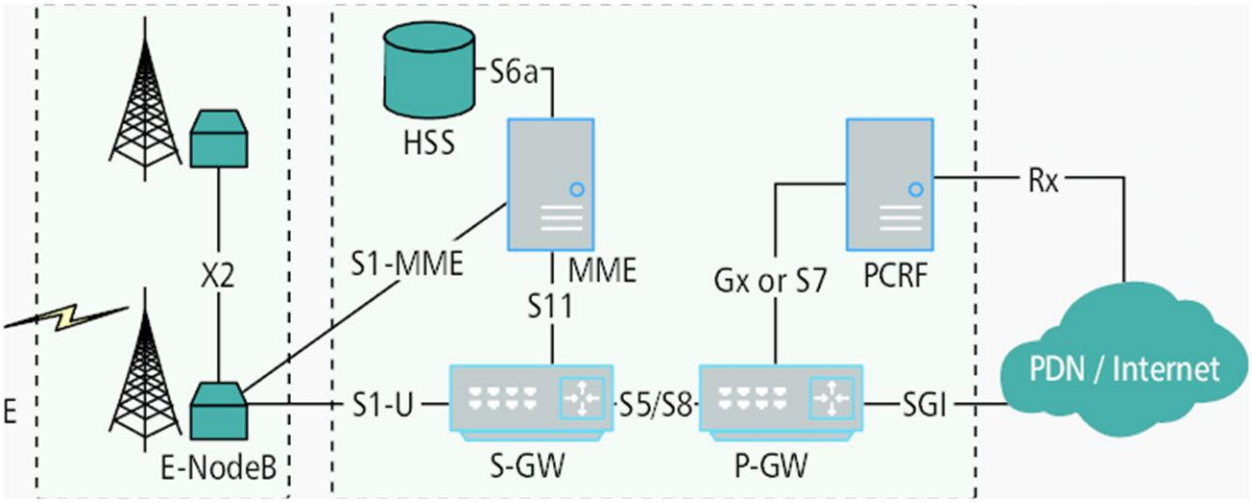
Synchronous Ethernet is an ITU-T standard that provides mechanisms to transfer frequency over the Ethernet physical layer, further made traceable to an external source such as a network clock. A Synchronization Status Message is sent over the Synchronous Ethernet. The SSM contains an indication of the quality level of the clock that is driving the synchronization chain. The Ethernet Synchronization Message Channel (ESMC) is used for propagation of the SSM through the Synchronous Ethernet network. When the SSM is not met with an adequate response arising due to latency or transmission problems an Alarm arises.

2. Base station synchronization problem

This Issue occurs when there is a difference between BTS master clock and the reference frequency. The affected RAT (Radio Access Technology) is **LTE.** When this problem arises time and frequency signals are deemed inaccurate and unstable leading to unsatisfactory performance.

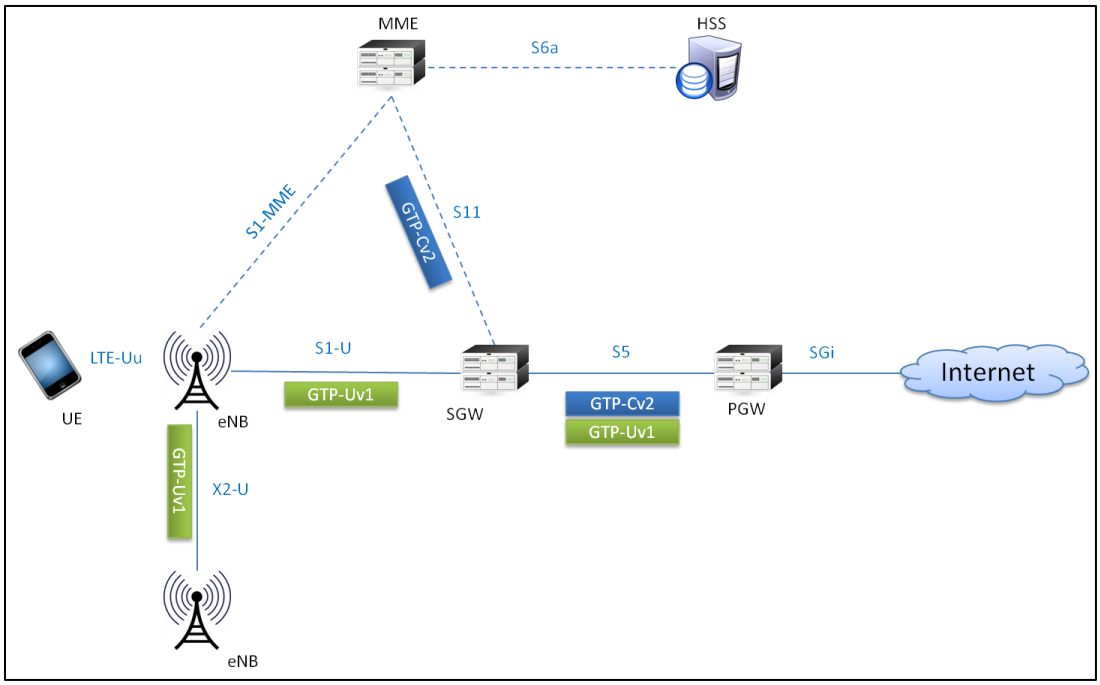
TNL FAILURES

Transport Network Layer failures mainly occur due to connection failure in the S1 Interface. It arises when the transport path for connection to the MME (Mobile Management Entity) is disconnected or broken. The S1 Interface is the interface which connects the E-UTRAN (Evolved UMTS Terrestrial Radio Network) and the EPC (Evolved Packet Core).



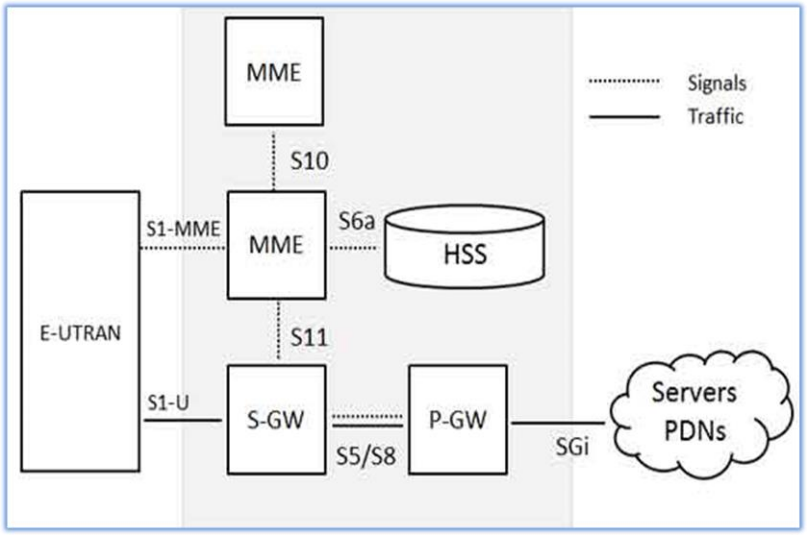
GTP-U PATH FAILURE

GTP-U is used for carrying user data within the GPRS core network and between the radio core network and the access network. The user data can be transported in packets of IPv4, IPv6, or PPP formats. A GTP-U peer may send an Echo Request on a path to the other GTP-U peer to find out if it is alive. Echo Request messages can be sent for each path in use. It will return positive if activated or will indicate a failure and give a reason for that failure.



MME CONNECTION FAILURE

The MME controls the high-level operation of the mobile by means of signalling messages and Home Subscriber Server (HSS). MME communication failure means that the LTE Extender is not able to maintain connectivity with servers over your home Internet connection.



3G TECHNOLOGY ALARMS

ET ERROR (ERR SEC)

The time during which Test Agent measured a high level of packet loss or delay is called Errored Second. ET errors are generated based on Err Sec. It is a very frequently occurring alarm which needs to be curbed.

SEVERELY ERROR SECONDS

This metric deals with Errored Seconds beyond a certain threshold.

SCTP FAILURE

Stream Control Transmission Protocol is a transport layer protocol that ensures reliable in-sequence transport of data. Faults can occur as an endpoint failure which leads to the SCTP Association being lost or as an SCTP Path failure.

2G TECHNOLOGY ALARMS

FRAME LOSS

If encountered, it means that a network device is not correctly configured or facing high traffic and is over-subscribed. It essentially is an implicit signal that a network is congested. Customers experiencing call drops is a prime example of this.

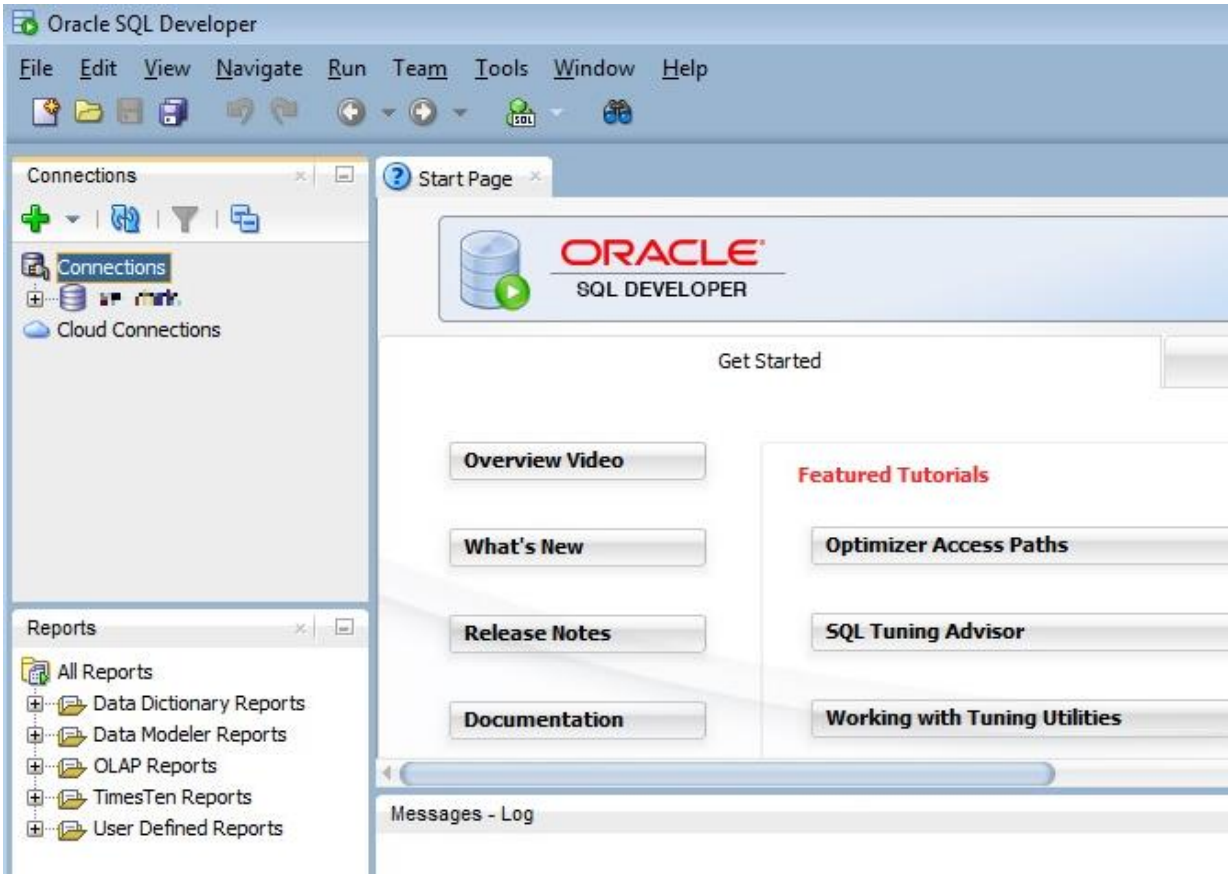
ORACLE DATABASE

Oracle database (Oracle DB) is a relational database management system (RDBMS) from the Oracle Corporation. It is one of the most trusted and widely-used relational database engines. The system is built around a relational database framework in which data objects may be directly accessed by users (or an application front end) through structured query language (SQL). Oracle is a fully scalable relational database architecture and is used by Vodafone Idea Limited. The Oracle database has its own network component to allow communications across networks.
A key feature of Oracle is that its architecture is split between the logical and the physical. This structure means that for large-scale distributed computing, also known as grid computing, the data location is irrelevant and transparent to the user, allowing for a more modular physical structure that can be added to and altered without affecting the activity of the database, its data or users. This allows for a robust system to be devised due to the absence of a single point at which a failure can bring down the database, ensuring that any failure would be local only.

OSS DATABASE

The Data related to Alarm Monitoring is extracted from the OSS Database. This Database consists of enterprise level-data which can be split into different tables based on functionality or usage. Alarm data can be filtered by different parameters such as E-Node_B name, severity, alarm number, start & cancel time and so on. This Process is manually done using the Oracle SQL Developer.

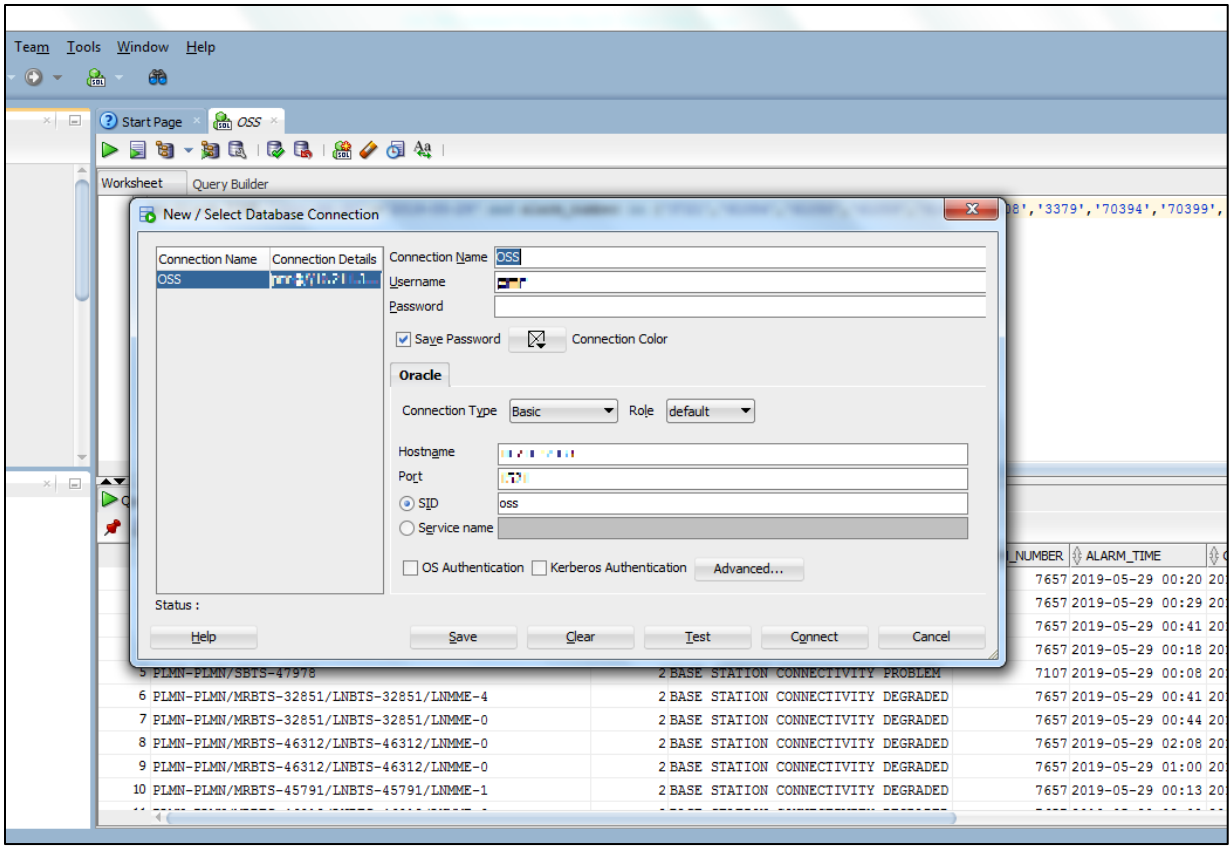
ORACLE SQL DEVELOPER

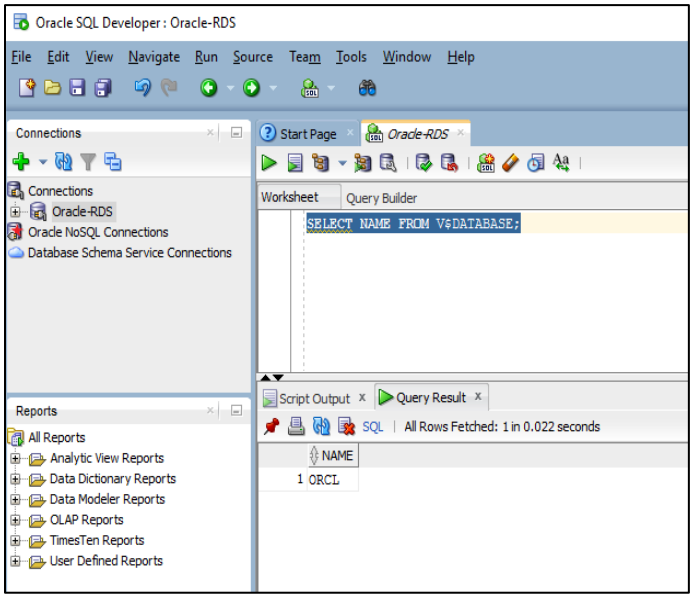


Oracle SQL Developer is a free graphical tool that enhances productivity and simplifies database development tasks using the Java Development Kit. Using SQL Developer, users can browse database objects, run SQL statements, edit and debug SQL statements and run reports, whether provided or created. users can connect to any supported Oracle Database, for all Oracle database editions including Express Edition.

To create a Database Connection click on the '+' sign on the left of the window near the Connections Label. To connect to a previously saved Database connection, simply double-click on the DB Name under Connections.

Enter the relevant details in the popup-tab and then click connect to establish a connection.





Enter the Query into the Worksheet and click RUN. The Query result is displayed on the bottom half on the window. This can be extracted to an excel file. This data can be further analysed and be used to prepare a report.

TOOLS USED FOR AUTOMATION

Oracle Instant Client

Oracle Instant Client enables applications to connect to a local or remote Oracle Database for development and production deployment. The Instant Client libraries provide the necessary network connectivity, as well as basic and high-end data features, to make full use of Oracle Database. It underlies the Oracle APIs of popular languages and environments including Node.js, Python and PHP. Tools included in Instant Client, such as SQL*Plus and Oracle Data Pump, provide quick and convenient data access. In this project, I Make use of the Oracle Instant Client and the cx_Oracle Python Module that enables access to Oracle Database and conforms to the Python database API specification.

CX_ORACLE

cx_Oracle is a Python extension module that enables access to Oracle Database. It conforms to the Python database API 2.0 specification with a considerable number of additions and a couple of exclusions.

Cx_Oracle can be installed using:

```
python -m pip install cx_Oracle --upgrade
```

A connection can be established using the connection Object :

```
import cx_Oracle
connection = cx_Oracle.connect("<username>/<password>@<hostIP>:<port>/<DB Name>”)
```

A cursor can be defined using the cursor() method of the connection object.

```
cursor = connection.cursor()
```

A database cursor is a control structure that can be used to traverse over records in a database. Cursors facilitate subsequent processing in conjunction with the traversal, such as retrieval, addition and removal of database records. The database cursor characteristic of traversal makes cursors akin to the programming language concept of an iterator. To interact with the database using Python we use the cursor object. This makes it possible to perform complex Logic operations on the Database.

PyQt-GUI

To enable the Python script to be implemented in a more user-friendly way, a Graphical User Interface was created using PyQt. PyQt is a python binding of the open-source widget-toolkit Qt, which also functions as a cross-platform application development framework. Qt is a popular C++ framework for writing GUI applications for all major desktop, mobile, and embedded platforms.

PANDAS

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured and time series data both easy and intuitive. It is a fundamental high-level building block for doing practical, real-world data analysis in Python. The analysis of Data extracted from the Oracle Database is done in the backend using the help of Pandas.

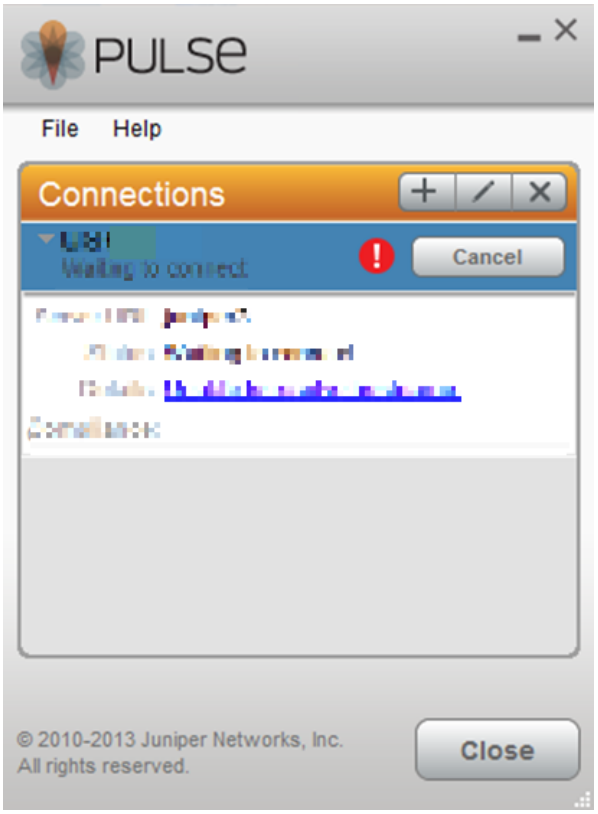
FOLIUM

Folium is a powerful Python library that can be used to create several types of Leaflet maps. It is a Python wrapper for a tool called leaflet.js. It enables us to create interactive maps with JS doing the work in the background

HTML & CSS

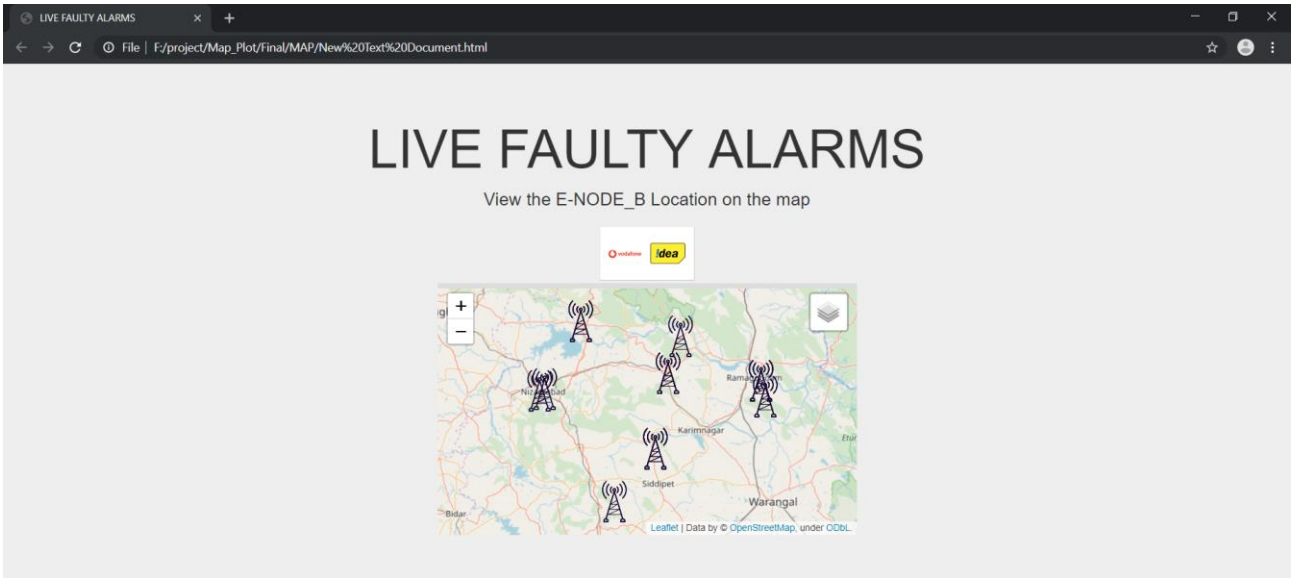
A simple webpage was styled using HTML with CSS. HTML stands for Hypertext Markup Language and CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media.

Juniper | Pulse Secure



Enterprises and service providers have the difficult challenge of providing location- and device-independent network connectivity that is secure and capable of controlling resource access for authorized users. Breaches and threats continue to spiral out of control, and increasing numbers of employees and users want to use their own personal mobile and computing devices to access enterprise data and applications, making this challenge even more difficult. Juniper Networks® Junos® Pulse Secure Access Service provides secure, authenticated access for remote and mobile users from any web-enabled device to corporate resources—anytime, anywhere. The Junos Pulse Secure Access Service is a leading, most widely deployed SSL VPN, and the remote access standard for organizations of any size, across every major industry.

LIVE FAULTY ALARM WEBPAGE



A Local Webpage was created using HTML with CSS. An Open Street Map was embedded onto the page with the Tower icons showing the location of each faulty eNodeB where the generated alarm had not been resolved.

The HTML tag – embed—can be used to embed an external application or interactive content

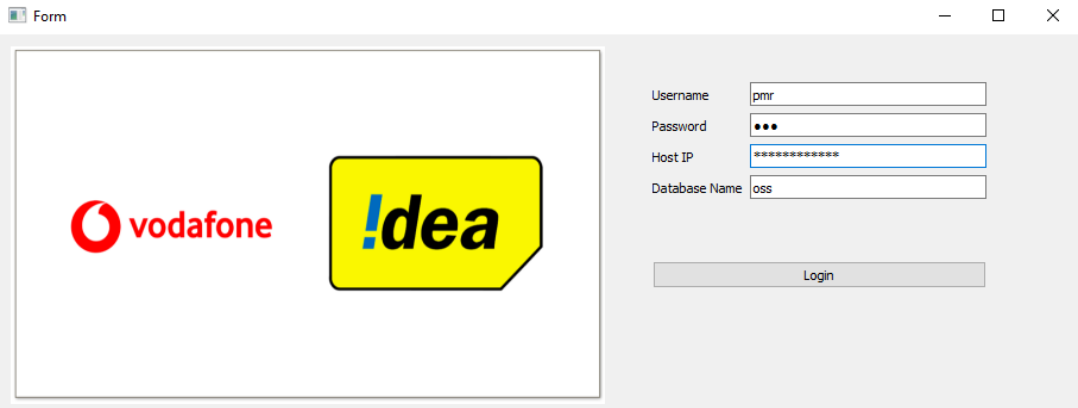
```
<embed src="map location/address" style="width:width in px; height: height in px;">
```

The three main steps to create the map are:

1. Connecting to the Oracle Database and Exporting the Data of eNodeB's where certain critical Alarms are unresolved. Unresolved Alarms can be identified by passing CANCEL_TIME as NULL in the query.
2. Analysis of the Extracted Data using Pandas and Combining it with the Coordinates Database.
3. Using Pandas, Folium, HTML and CSS to Map the relevant data onto the HTML Web Page.

ALARM DATA APPLICATION

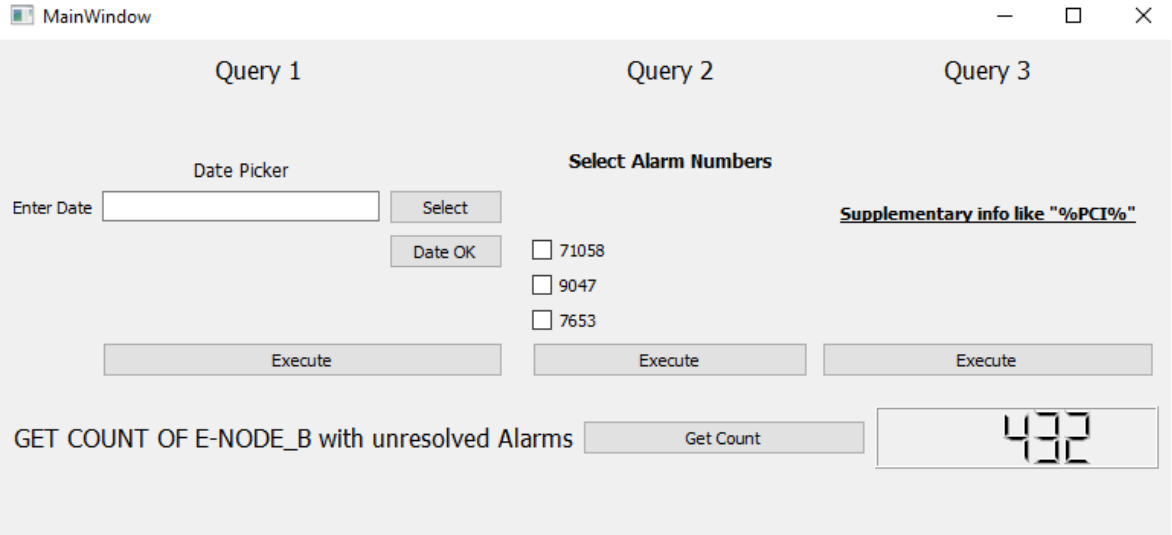
To automate the daily Alarm Monitoring reports, a simple application was created using Python. The script was written with the aid of cx_Oracle and Pandas libraries. The Graphical User Interface was created using PyQt5. The aim of this mini-application was to easily automate certain time-consuming procedures of Alarm Monitoring. Based on the User's interaction with the application, the required tasks are taken care of in the background and the final data is stored in an excel file with a date-stamp.



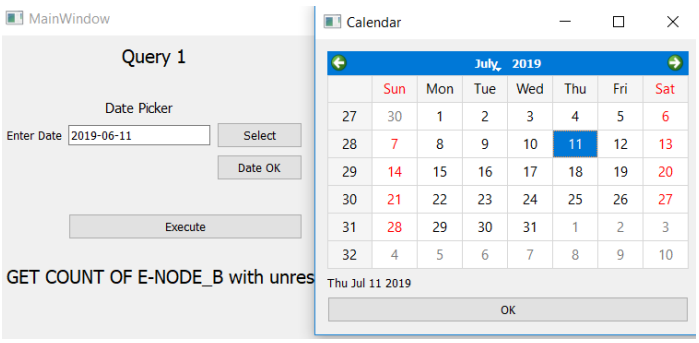
CREDENTIAL CHECK/LOGIN PAGE

The first window prompts the user to enter the details to establish a connection with the database. The data inputted in parsed and then passed to the backend Python script in which it is passed to the connect() method of cx_Oracle. On successfully connecting to the database, a new window pops up with an interface that allows the user to perform different tasks.

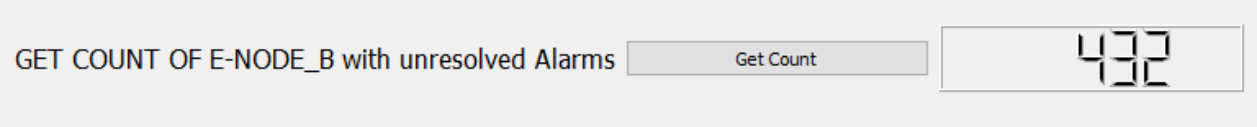
MAIN WINDOW



The user can perform 4 different tasks using the various button functionalities provided. Query 1, Query 2 and Query 3 are associated with various specific tasks which are required to be performed. The variable parameter required to perform Query 1 is a date variable which can be selected from the Calendar options provided in the Select Button. On Execution, the required task is performed and the final necessary output is provided in an Excel File named in the <file_name>+<date_stamp> form.



In Query 2, the user can select the required critical alarm numbers, which is passed to the backend script which makes sure that the required procedure is carried out without any hassle and the expected output is provided in an Excel File named in the <file_name>+<date_stamp> form. Query 3 has a rather simple functionality. It parses the data and extracts rows from the fx_alarm table while Supplementary info contains “PCI”. The required export is stored in the same name format as the previous tasks.



The bottom Get Count button can be used to see how many eNodeB have at least one, unresolved live alarm. Live Alarms have their CANCEL_TIME set to NULL. The final count can be seen on the LCD-like number display.

CONCLUSION

From this study, I was able to learn a lot about the Alarms in Telecommunications. I learned how these affect the network and was able to monitor them. Automation of certain procedures helps to make the process more efficient. Graphical representation of the location of eNodeB's with unresolved live alarms provides a better understanding of the problem. I was unable to acquire a basic understanding of Web-development by making the live alarm page. To create a Interface to achieve these procedures I had to learn Data Analysis create a backend script using Python. This allowed me to build my Computer Programming skills. Fault management systems uses complex filtering systems to assign alarms to severity levels. Understanding this dynamic was key to my work here as Alarm Monitoring is an integral part of ensuring sustained maintenance and improvement in the quality of the network.

APPENDIX

(The end results of the following code were only in the alpha testing mode, further testing could not be continued due to company protocol resulting in inability to iron out bugs present.)

test1.py

```
from PyQt5 import QtCore, QtGui, QtWidgets
import back
from mainwin import Ui_MainWindow

class Ui_Form(object):
    def ok (self):
        print("ty")
        self.window = QtWidgets.QMainWindow()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self.window)
        self.window.show()


    def message(self,title,des):
        message = QtWidgets.QMessageBox()
        message.setWindowTitle(title)
        #message.setWindowIcon(QtWidgets.QMessageBox.Information)
        message.setText(des)

        message.exec_()

    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(910, 318)
        self.formLayoutWidget = QtWidgets.QWidget(Form)
        self.formLayoutWidget.setGeometry(QtCore.QRect(550, 40, 281, 221))
        self.formLayoutWidget.setObjectName("formLayoutWidget")
        self.formLayout = QtWidgets.QFormLayout(self.formLayoutWidget)
        self.formLayout.setContentsMargins(0, 0, 0, 0)
        self.formLayout.setObjectName("formLayout")
        self.user_name = QtWidgets.QLabel(self.formLayoutWidget)
        self.user_name.setObjectName("user_name")
        self.formLayout.setWidget(1, QtWidgets.QFormLayout.LabelRole, self.user_name)
        self.username_input = QtWidgets.QLineEdit(self.formLayoutWidget)
        self.username_input.setObjectName("username_input")
        self.username_input.setText("hr")
```

```
        self.formLayout.setWidget(1, QtWidgets.QFormLayout.FieldRole, self.username_input)
        self.password = QtWidgets.QLabel(self.formLayoutWidget)
        self.password.setObjectName("pass")
        self.formLayout.setWidget(2, QtWidgets.QFormLayout.LabelRole, self.password)
        self.pass_input = QtWidgets.QLineEdit(self.formLayoutWidget)
        self.pass_input.setObjectName("pass_input")
        self.pass_input.setEchoMode(QtWidgets.QLineEdit.Password)
        #self.pass_input.setText("hr")
        self.formLayout.setWidget(2, QtWidgets.QFormLayout.FieldRole, self.pass_input)
        self.ip_add = QtWidgets.QLabel(self.formLayoutWidget)
        self.ip_add.setObjectName("ip_add")
        self.formLayout.setWidget(3, QtWidgets.QFormLayout.LabelRole, self.ip_add)
        self.ip_input = QtWidgets.QLineEdit(self.formLayoutWidget)
        self.ip_input.setObjectName("ip_input")
        self.ip_input.setText("localhost")
        self.formLayout.setWidget(3, QtWidgets.QFormLayout.FieldRole, self.ip_input)
        self.db_name = QtWidgets.QLabel(self.formLayoutWidget)
        self.db_name.setObjectName("db_name")
        self.formLayout.setWidget(4, QtWidgets.QFormLayout.LabelRole, self.db_name)
        self.db_input = QtWidgets.QLineEdit(self.formLayoutWidget)
        self.db_input.setObjectName("db_input")
        self.db_input.setText("orcl")
        self.formLayout.setWidget(4, QtWidgets.QFormLayout.FieldRole, self.db_input)
        self.login_Button = QtWidgets.QPushButton(self.formLayoutWidget)
        self.login_Button.setObjectName("login_Button")
        self.login_Button.clicked.connect(self.login_click)
        self.formLayout.setWidget(7, QtWidgets.QFormLayout.SpanningRole, self.login_Button)
        spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
                                           QtWidgets.QSizePolicy.Expanding)
        self.formLayout.setItem(5, QtWidgets.QFormLayout.SpanningRole, spacerItem)
        self.voda_image = QtWidgets.QLabel(Form)
        self.voda_image.setEnabled(True)
        self.voda_image.setGeometry(QtCore.QRect(10, 10, 500, 300))
        self.voda_image.setText("")
        self.voda_image.setPixmap(QtGui.QPixmap("voda.png"))
        self.voda_image.setScaledContents(True)
        self.voda_image.setTextInteractionFlags(QtCore.Qt.NoTextInteraction)
        self.voda_image.setObjectName("voda_image")

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.user_name.setText(_translate("Form", "Username"))
```

```
        self.password.setText(_translate("Form", "Password"))
        self.ip_add.setText(_translate("Form", "Host IP"))
        self.db_name.setText(_translate("Form", "Database Name"))
        self.login_Button.setText(_translate("Form", "Login"))

    def login_click(self):

        tes1, tes2, tes3, tes4 = self.username_input.text(), self.pass_input.text(),\
                            self.ip_input.text(),self.db_input.text()

        if back.cred_check(tes1,tes2,tes3,tes4):
            print ("hi")
            #self.message("Credential Check","Connection Successful")
            print ("h00000i")
            '''
            self.welcomeWindow =  QtGui.QMainWindow()
            self.ui= Ui_MainWindow()
            self.setupUi(self.welcomeWindow)
            self.welcomeWindow.show()
            '''
            self.ok()
            print("byr")


        else:
            self.message("Credential Check","Connection Unsuccessful")




if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

mainwin.py

```
from PyQt5 import QtCore, QtGui, QtWidgets
import matchlatlong
from cal import Example
from time import strptime


global from_d, to_d

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(774, 334)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.verticalLayoutWidget = QtWidgets.QWidget(self.centralwidget)
        self.verticalLayoutWidget.setGeometry(QtCore.QRect(350, 10, 181, 211))
        self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.verticalLayoutWidget)
        self.verticalLayout.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout.setObjectName("verticalLayout")
        self.label_5 = QtWidgets.QLabel(self.verticalLayoutWidget)
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label_5.setFont(font)
        self.label_5.setAlignment(QtCore.Qt.AlignCenter)
        self.label_5.setObjectName("label_5")
        self.verticalLayout.addWidget(self.label_5)
        spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout.addItem(spacerItem)
        self.label_6 = QtWidgets.QLabel(self.verticalLayoutWidget)
        font = QtGui.QFont()
        font.setPointSize(9)
        font.setBold(True)
        font.setWeight(75)
        self.label_6.setFont(font)
        self.label_6.setAlignment(QtCore.Qt.AlignCenter)
        self.label_6.setObjectName("label_6")
        self.verticalLayout.addWidget(self.label_6)
        spacerItem1 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout.addItem(spacerItem1)
        self.an_71058 = QtWidgets.QCheckBox(self.verticalLayoutWidget)
        self.an_71058.setObjectName("an_71058")
```

```
        self.verticalLayout.addWidget(self.an_71058)
        self.an_9047 = QtWidgets.QCheckBox(self.verticalLayoutWidget)
        self.an_9047.setObjectName("an_9047")
        self.verticalLayout.addWidget(self.an_9047)
        self.an_7653 = QtWidgets.QCheckBox(self.verticalLayoutWidget)
        self.an_7653.setObjectName("an_7653")
        self.verticalLayout.addWidget(self.an_7653)

        self.q2_execute = QtWidgets.QPushButton(self.verticalLayoutWidget)
        self.q2_execute.setObjectName("q2_execute")
        self.q2_execute.clicked.connect(self.three_alarms)

        self.verticalLayout.addWidget(self.q2_execute)
        self.verticalLayoutWidget_2 = QtWidgets.QWidget(self.centralwidget)
        self.verticalLayoutWidget_2.setGeometry(QtCore.QRect(540, 10, 218, 211))
        self.verticalLayoutWidget_2.setObjectName("verticalLayoutWidget_2")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.verticalLayoutWidget_2)
        self.verticalLayout_2.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.label_8 = QtWidgets.QLabel(self.verticalLayoutWidget_2)
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label_8.setFont(font)
        self.label_8.setAlignment(QtCore.Qt.AlignCenter)
        self.label_8.setObjectName("label_8")
        self.verticalLayout_2.addWidget(self.label_8)
        spacerItem2 = QtWidgets.QSpacerItem(188, 38, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout_2.addItem(spacerItem2)
        self.label_9 = QtWidgets.QLabel(self.verticalLayoutWidget_2)
        font = QtGui.QFont()
        font.setBold(True)
        font.setUnderline(True)
        font.setWeight(75)
        self.label_9.setFont(font)
        self.label_9.setAlignment(QtCore.Qt.AlignCenter)
        self.label_9.setObjectName("label_9")
        self.verticalLayout_2.addWidget(self.label_9)
        spacerItem3 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout_2.addItem(spacerItem3)

        self.q2_execute_2 = QtWidgets.QPushButton(self.verticalLayoutWidget_2)
        self.q2_execute_2.setObjectName("q2_execute_2")
        self.q2_execute_2.clicked.connect(self.pci_click)
```

```
        self.verticalLayout_2.addWidget(self.q2_execute_2)
        self.gridLayoutWidget = QtWidgets.QWidget(self.centralwidget)
        self.gridLayoutWidget.setGeometry(QtCore.QRect(10, 10, 321, 211))
        self.gridLayoutWidget.setObjectName("gridLayoutWidget")
        self.gridLayout = QtWidgets.QGridLayout(self.gridLayoutWidget)
        self.gridLayout.setContentsMargins(0, 0, 0, 0)
        self.gridLayout.setObjectName("gridLayout")
        spacerItem4 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.gridLayout.addItem(spacerItem4, 2, 0, 1, 1)
        '''
        self.label_2 = QtWidgets.QLabel(self.gridLayoutWidget)
        self.label_2.setObjectName("label_2")
        self.gridLayout.addWidget(self.label_2, 5, 0, 1, 1)
        '''
        self.label = QtWidgets.QLabel(self.gridLayoutWidget)
        self.label.setObjectName("label")
        self.gridLayout.addWidget(self.label, 4, 0, 1, 1)

        self.to_btn = QtWidgets.QPushButton(self.gridLayoutWidget)
        self.to_btn.setObjectName("to_btn")
        self.to_btn.clicked.connect(self.datok)

        self.gridLayout.addWidget(self.to_btn, 5, 2, 1, 1)

        self.q1_execute = QtWidgets.QPushButton(self.gridLayoutWidget)
        self.q1_execute.setObjectName("q1_execute")
        self.q1_execute.clicked.connect(self.q1_exec)

        self.gridLayout.addWidget(self.q1_execute, 7, 1, 1, 2)
        self.label_3 = QtWidgets.QLabel(self.gridLayoutWidget)
        font = QtGui.QFont()
        font.setPointSize(9)
        self.label_3.setFont(font)
        self.label_3.setAlignment(QtCore.Qt.AlignCenter)
        self.label_3.setObjectName("label_3")
        self.gridLayout.addWidget(self.label_3, 3, 1, 1, 1)

        self.from_date_preview = QtWidgets.QLineEdit(self.gridLayoutWidget)
        self.from_date_preview.setObjectName("from_date_preview")


        self.gridLayout.addWidget(self.from_date_preview, 4, 1, 1, 1)
        spacerItem5 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.gridLayout.addItem(spacerItem5, 6, 1, 1, 1)
```

```
'''
self.to_date_preview = QtWidgets.QLineEdit(self.gridLayoutWidget)
self.to_date_preview.setObjectName("to_date_preview")

self.gridLayout.addWidget(self.to_date_preview, 5, 1, 1, 1)
'''
self.from_btn = QtWidgets.QPushButton(self.gridLayoutWidget)
self.from_btn.setObjectName("from_btn")
self.from_btn.clicked.connect(self.from_cal)

self.gridLayout.addWidget(self.from_btn, 4, 2, 1, 1)
self.label_4 = QtWidgets.QLabel(self.gridLayoutWidget)
font = QtGui.QFont()
font.setPointSize(12)
self.label_4.setFont(font)
self.label_4.setAlignment(QtCore.Qt.AlignCenter)
self.label_4.setObjectName("label_4")
self.gridLayout.addWidget(self.label_4, 1, 0, 1, 3)
self.horizontalLayoutWidget = QtWidgets.QWidget(self.centralwidget)
self.horizontalLayoutWidget.setGeometry(QtCore.QRect(10, 240, 751, 41))
self.horizontalLayoutWidget.setObjectName("horizontalLayoutWidget")
self.horizontalLayout = QtWidgets.QHBoxLayout(self.horizontalLayoutWidget)
self.horizontalLayout.setContentsMargins(0, 0, 0, 0)
self.horizontalLayout.setObjectName("horizontalLayout")
self.label_7 = QtWidgets.QLabel(self.horizontalLayoutWidget)
font = QtGui.QFont()
font.setPointSize(12)
font.setUnderline(False)
self.label_7.setFont(font)
self.label_7.setMouseTracking(True)
self.label_7.setObjectName("label_7")
self.horizontalLayout.addWidget(self.label_7)

self.q1_execute_2 = QtWidgets.QPushButton(self.horizontalLayoutWidget)
self.q1_execute_2.setObjectName("q1_execute_2")
self.q1_execute_2.clicked.connect(self.faulty_count_click)

self.horizontalLayout.addWidget(self.q1_execute_2)

self.lcdNumber = QtWidgets.QLCDNumber(self.horizontalLayoutWidget)
self.lcdNumber.setObjectName("lcdNumber")
#lid=self.disp_lcd()
lid=432
self.lcdNumber.display(lid)

self.horizontalLayout.addWidget(self.lcdNumber)
```

```
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 774, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.label_5.setText(_translate("MainWindow", "Query 2"))
        self.label_6.setText(_translate("MainWindow", "Select Alarm Numbers"))
        self.an_71058.setText(_translate("MainWindow", "71058"))
        self.an_9047.setText(_translate("MainWindow", "9047"))
        self.an_7653.setText(_translate("MainWindow", "7653"))
        self.q2_execute.setText(_translate("MainWindow", "Execute"))
        self.label_8.setText(_translate("MainWindow", "Query 3"))
        self.label_9.setText(_translate("MainWindow", "Supplementary info like \"%PCI%\""))
        self.q2_execute_2.setText(_translate("MainWindow", "Execute"))
        #self.label_2.setText(_translate("MainWindow", "Date OK"))
        self.label.setText(_translate("MainWindow", "Enter Date"))
        self.to_btn.setText(_translate("MainWindow", "Date OK"))
        self.q1_execute.setText(_translate("MainWindow", "Execute"))
        self.label_3.setText(_translate("MainWindow", "Date Picker"))
        self.from_btn.setText(_translate("MainWindow", "Select"))
        self.label_4.setText(_translate("MainWindow", "Query 1"))
        self.label_7.setText(_translate("MainWindow", "GET COUNT OF E-NODE_B with
unresolved Alarms"))
        self.q1_execute_2.setText(_translate("MainWindow", "Get Count"))

    def disp_lcd(self):
        file = open("fault_count.txt", mode='r')

        num=file.read()
        file.close()
        return num

    def three_alarms():
        kk

    def from_cal(self):
```

```
        print("ty")
        self.window = QtWidgets.QWidget()
        self.ui = Example()
        self.ui.initUI()
        self.window.show()

    def datok(self):
        date=date_text()
        self.from_date_preview.setText(date)


    def faulty_count_click(self):
        fal_count=matchlatlong.count_faulty()
        file = open("fault_count.txt", mode='w')
        file.write('%s' %fal_count)
        file.close()

        print (fal_count)
        #return fal_count



    def pci_click(self):
        matchlatlong.supp_info_pci()




    def q1_exec(self):

        res1 = self.from_date_preview.text()
        matchlatlong.all_now(res1)



def date_text():
    file = open("from.txt", mode='r')

    with open('from.txt') as f:
         list=[word for line in f for word in line.split()]
    mon=list[1]
    mon = strptime(str(mon),'%b').tm_mon
    if mon<10:
        mon = '0' + str(mon)
```

```
    date=list[3]+'-'+mon +'-'+ list[2]
    file.close()
    print (date)


    return date


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    ui = Ui_MainWindow()
    MainWindow = QtWidgets.QMainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

cal.py

```
from PyQt5.QtWidgets import (QWidget, QCalendarWidget,
    QLabel, QApplication, QVBoxLayout,QPushButton)
from PyQt5.QtCore import QDate
import sys

global ap
class Example(QWidget):

    def __init__(self):
        super().__init__()

        self.initUI()


    def initUI(self):

        vbox = QVBoxLayout(self)
        ap=''

        cal = QCalendarWidget(self)
        cal.setGridVisible(True)
        cal.clicked[QDate].connect(self.showDate)

        vbox.addWidget(cal)

        self.lbl = QLabel(self)
        date = cal.selectedDate()
```

```
        self.lbl.setText(date.toString())

        self.ok = QPushButton(self)
        self.ok.setText("OK")
        self.ok.clicked.connect(self.okay)




        vbox.addWidget(self.lbl)
        vbox.addWidget(self.ok)

        self.setLayout(vbox)

        self.setGeometry(300, 300, 350, 300)
        self.setWindowTitle('Calendar')
        self.show()


    def showDate(self, date):

        self.lbl.setText(date.toString())
        #nonlocal ap
        ap= date.toString()
        file = open("from.txt", mode='w')
        file.write('%s' %ap)
        file.close()

        print(ap)

    def okay(self):



        self.close()



if __name__ == '__main__':

    app = QApplication(sys.argv)

    #doot = QWidget()

    ex = Example()
```

```
    #ex.initUI(doot)
    #ex.show()
    sys.exit(app.exec_())
```

back.py

```
import cx_Oracle


def cred_check(a,b,c,d):
        apple = a+'/'+b+'@'+c+'/'+d

        conn=cx_Oracle.connect(apple)
        if conn :
            return 1 # returns 1 if credentials work
        else :
            return 0 # returns 0 if credentials don't work

        conn.close()
```

matchlatlong.py

```
import cx_Oracle,time,datetime,folium
from datetime import datetime as dt
import pandas


def count_faulty():
    conn=cx_Oracle.connect('pmr/pmr@10.211.32.133/oss')
    cur=conn.cursor()
    query = "select count(unique DN) from fx_alarm where Text like '%CELL FAULTY%' and
CANCEL_TIME is null"
    cur.execute(query)
    faulty_count = cur.fetchall()
    cur.close()
    conn.close()
    return faulty_count


def faulty_live():
    conn=cx_Oracle.connect('pmr/pmr@10.211.32.133/oss')
    cur=conn.cursor()
    print("hi")
```

```
    query = "select unique DN from fx_alarm where Text like '%CELL FAULTY%' and
CANCEL_TIME is null"
    cur.execute(query)
    faulty_live_list = cur.fetchall()
    cur.close()
    conn.close()
    print(type(faulty_live_list[0][0]))


    return faulty_live_list

def cord_link_lat(dn_short):
    names=['FDD_E-NodeB','LAT','LONG']
    cord_book=pandas.read_excel("Book1.xls",header=None,names=names)

    book_dn = list(cord_book['FDD_E-NodeB'])
    book_lat = list(cord_book["LAT"])
    book_long = list(cord_book["LONG"])



    for (j,test_single) in enumerate(book_dn):

        if (str(dn_short)) == (str(test_single)):
            import math

            if math.isnan(book_lat[j]):
                pass

            elif math.isnan(book_long[j]):
                pass
            else:
                return str(book_lat[j])
                break #breakhere


def cord_link_long(dn_short):
    names=['FDD_E-NodeB','LAT','LONG']
    cord_book=pandas.read_excel("Book1.xls",header=None,names=names)

    book_dn = list(cord_book['FDD_E-NodeB'])
    book_lat = list(cord_book["LAT"])
    book_long = list(cord_book["LONG"])
```

```
    for (j,test_single) in enumerate(book_dn):

        if (str(dn_short)) == (str(test_single)):
            import math

            if math.isnan(book_lat[j]):
                pass

            elif math.isnan(book_long[j]):
                pass
            else:
                print(book_lat[j])
                return str(book_long[j])
                break #breakhere



def shrink_dn (faulty_live_list):
   # df_faulty= pandas.DataFrame.from_records(faulty_live_list)
    dn=faulty_live_list

    DN_small = pandas.Series([])
    Long = pandas.Series(['0'])
    Lat = pandas.Series(['0'])


    for (j,dn_single) in enumerate(dn):
        dn_short=""
        for (i,dn_single_alpha) in enumerate(dn_single[0]):


            if dn_single_alpha.isdigit():

                dn_short = dn_short + dn_single_alpha

            if i==21:


                break #breakhere

        DN_small [j] = dn_short



        b=cord_link_lat(str(dn_short))
        a=cord_link_long(str(dn_short))
```

```
        Lat[j]=b
        Long[j]=a


    df_faulty = pandas.DataFrame(list(zip(DN_small, Long,Lat)),
            columns =['DN', 'LONG','LAT'])




    return df_faulty



def plot_on_map(df_faulty):
    map_dn=list(df_faulty["DN"])
    map_lat=list(df_faulty["LAT"])
    map_long=list(df_faulty["LONG"])


    map = folium.Map(location=[18.2340,78.703],zoom_start=8, titles="Stamen")

    fg_node=folium.FeatureGroup(name="Node")
    #fgpop=folium.FeatureGroup(name="Population")

    for lt,ln,dn in zip(map_lat,map_long,map_dn):

       if lt is not None:
           if ln is not None:

              icon = folium.features.CustomIcon('tower.png',icon_size=(50, 50))
              print(lt)
              fg_node.add_child(folium.Marker(location=[float(str(lt)),float(str(ln))],
                                    radius=6,popup=str(dn)+"m",  color = 'grey',
                                    fill_opacity=0.7,icon=icon))

    map.add_child(fg_node)
    map.add_child(folium.LayerControl())
    map.save("new.html")


def supp_info_pci(name):
    conn=cx_Oracle.connect('pmr/pmr@10.211.32.133/oss')
    cur=conn.cursor()
    query = "select * from fx_alarm where SUPPLEMENTARY_INFO like '%PCI%' "
    cur.execute(query)
```

```
    pci = cur.fetchall()

    pci_export = pandas.DataFrame.from_records(pci)
    date=str(dt.now().year) + '-' + str(dt.now().month) + '-' + str(dt.now().day)
    export_name= 'export'+ date + '.xlsx'
    pci_export.to_excel(export_name,encoding="utf-8")


    cur.close()
    conn.close()

def all_now(date):
    conn=cx_Oracle.connect('pmr/pmr@10.211.32.133/oss')
    cur=conn.cursor()
    query=" Select dn,SEVERITY ,text,alarm_number,to_char(alarm_time,'yyyy-mm-dd
hh24:mi')\
            Alarm_time,to_char(cancel_time,'yyyy-mm-dd hh24:mi') Cancel_time,
SUPPLEMENTARY_INFO from \
            fx_alarm where to_char(Alarm_TIME,'yyyy-mm-dd')="'+ date+"' and alarm_number
in \

('61611','3721','61084','61058','61059','61623','7108','3379','70394','70399','3159','7107','7657') "


    cur.execute(query)
    all = cur.fetchall()
    all_export = pandas.DataFrame.from_records(all)
    date=str(dt.now().year) + '-' + str(dt.now().month) + '-' + str(dt.now().day)
    export_name= 'export'+ date + '.xlsx'
    all_export.to_excel(export_name,encoding="utf-8")
    cur.close()
    conn.close()
```

webpage.html

```
<!DOCTYPE html>
<html>
<head>
  <title>LIVE FAULTY ALARMS</title>
  <link rel="stylesheet" type="text/css" href ="style.css">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>

</head>
<body>
  <div class="jumbotron text-center">
    <h1>LIVE FAULTY ALARMS</h1>
    <p>View the E-NODE_B Location on the map</p>
    <a href =""><img class="logo" src = "voda.png" height="200" alt="Voda_logo" ></a>


    <div class="container">
      <embed src="new.html" style="width:500px; height: 300px;">

      </div>
      <div class="container">
      </div>
  </div>
</body>
</html>
```

style.css

```
*{
    margin : 0;
    padding: 0;
}
.header
{
    height: 100vh;

}
.logo
{
    height : 70px;
    padding : 2px 10px;
```

```
}

.navbar-style
{
   box-shadow : 0 5px 10px #efefef;
}
.center {
  display: block;
  margin-left: auto;
  margin-right: auto;
  width: 50%;
}
```

livealarm.py

```
from matchlatlong import *

list=faulty_live()
list_short_cord= shrink_dn(list)
plot_on_map(list_short_cord)
```

Book1.xls

It contains the confidential company data of the coordinates parsed from the Oracle database.

Tower.png

voda.png

REFERENCES

- Understanding Two-Way Active Measurement Protocol on Routers ~ Retrieved from www.juniper.net
- LTE – ETSI ~ Retrieved from www.etsi.org
- Alarms – Nokia! ~ Retrieved from infocenter.nokia.com
- ASR 5000 Series Troubleshooting GTPC and GTPU and associated path failures ~ Retrieved from www.cisco.com
- The LTE Network Architecture ~ Retrieved from www.cse.unt.edu
- Synchronization of Telecommunication Networks ~ Retrieved from citeseerx.ist.psu.edu
- Oracle Python Developer Center ~ Retrieved from www.oracle.com
- PyQt5 ~ Retrieved from zetcode.com
- Web Development ~ Retrieved from www.techopedia.com
- How I Understood: Building Interactive Maps using Python, Leaflet.js and Folium ~ Retrieved from codeburst.io