



Software Requirements Specification

for

CDR ANALYSIS SYSTEM

Version 1.0

Prepared by

**AKSHIT AGARWAL
VISHESH VERMA
TARIT KANDPAL**

Mentor:

REVISIONS		III
1	INTRODUCTION	1
1.1	DOCUMENT PURPOSE	1
1.2	PRODUCT SCOPE	1
1.3	DOCUMENT CONVENTIONS	1
1.4	REFERENCES AND ACKNOWLEDGMENTS	2
2	OVERALL DESCRIPTION	3
2.1	PRODUCT PERSPECTIVE	3
2.2	PRODUCT FUNCTIONALITY	3
2.3	OPERATING ENVIRONMENT	3
2.4	DESIGN AND IMPLEMENTATION CONSTRAINTS	4
2.5	USER DOCUMENTATION	4
2.6	ASSUMPTIONS AND DEPENDENCIES	4
3	SPECIFIC REQUIREMENTS	5
3.1	EXTERNAL INTERFACE REQUIREMENTS	5
4	OTHER NON-FUNCTIONAL REQUIREMENTS	7
4.1	PERFORMANCE REQUIREMENTS	7
4.2	SOFTWARE QUALITY ATTRIBUTES	7
APPENDIX A – GROUP LOG		9
SOURCE CODE		10

1.1 Document Purpose

Our SRS is the first deliverable for the software project CDR Analysis System. Our SRS specifies all the features including technical requirements and interface requirements. It

enables the costing and pricing of the project. It enables the developers to convert the requirements directly to a technical design. It ensures that customer's expectations don't change during software development. If they do so SRS can be modified and costing can be done again on the changes required. This SRS contains Project Scope Section, Functional requirements, Analysis Models, External Interface Requirements and Non Functional requirements. Each of these have been explained below. The aim of this document is to gather, analyze and give an in-depth insight of the complete CDR Analysis System by defining the problem statement in detail.

1.2 Product Scope

This section includes a brief overview of the project and indicates the goals of this project and indicates the goals of this project including its benefits.

This software provides a platform to save all the CDRs from the last 3 months in a common database and then process queries according to the user requirements. This software is a great way to access and view the common parameters required for call analysis.

1.3 Document Conventions

This document follows the IEEE formatting requirements. We have used Arial font size 12 throughout the document for text. Document text is single spaced and maintain the 1" margins found in this template. Sub Headings have been written in Arial 14.

1.4 References and Acknowledgments

1. <https://stackoverflow.com>
2. <http://www.good-tutorials.com/>
3. <https://www.geeksforgeeks.org>
4. *Learn Python the Hard Way* by Zed A SHAW

2 Overall Description

2.1 Product Perspective

This CDR analysis software e provides a simple mechanism for both Analysts and server uploaders. Any Analyst can view data by applying queries of his/her choice on any of the parameters specified. The Uploader can insert new CDR data to the database according to the data recycle cycle..

2.2 Product Functionality

1. SERVER UPLOADER

1. The user can insert data into the database.

2. CDR ANALYST.

1. The user can login into his/her account.
2. The user can view all the data stored for the past 3 months.
3. The user can view data according to the default parameters
4. The user can process queries on the data selecting one or more queries

2.3 Operating Environment

The operating environment for the “Social media software” is as listed below-

Processor- Intel Core Xeon
Hard Disk-at least 100GB
Ram-16GB or higher
Libraries-MySQL.connector, Tkinter,OS
PyCharm Community 2019.1
MariaDB 4.01
Operating System- Ubuntu 16.04
Python 3.7

2.4 Design and Implementation Constraints

Memory- The VM should have minimum of 16GB RAM and 100GB hard-disk to successfully display 9 parameters of the whole CDR data.

Language-The user should know English to use the software

Login Page- The user should login using the ID specified by the developer.

Libraries-MySQL.connector, Tkinter,OS and Python are required for the software to run.

Security consideration-The user should remember always remember the IP address to access the software

2.5 User Documentation

A user manual would be provided explaining all the functionalities pertaining to the user. It would also include all the IP addresses and the login details required to access the software.

2.6 Assumptions and Dependencies

It is assumed that the user knows English..

It is assumed that the analyst knows he can only view data from the database. He has no authority to modify data.

It is assumed that the uploader knows that he can only insert data into the databse.

3 Specific Requirements

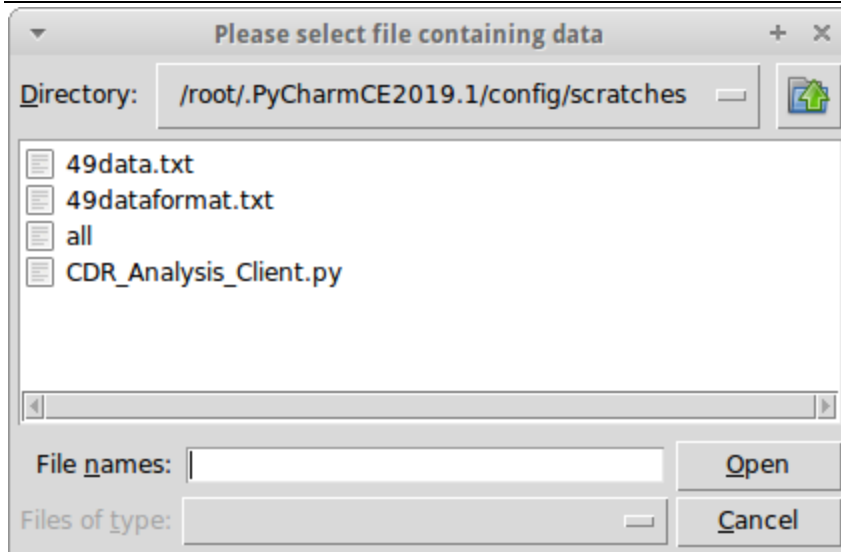
3.1 External Interface Requirements

3.1.1 User Interfaces

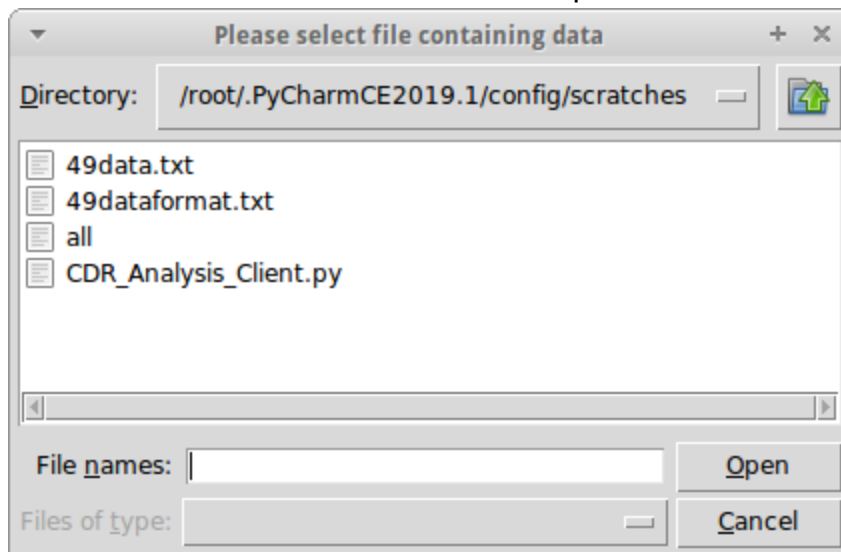
The interface for users is very user-friendly.

SERVER:

The opening screen provides a browser to select data format for uploading.

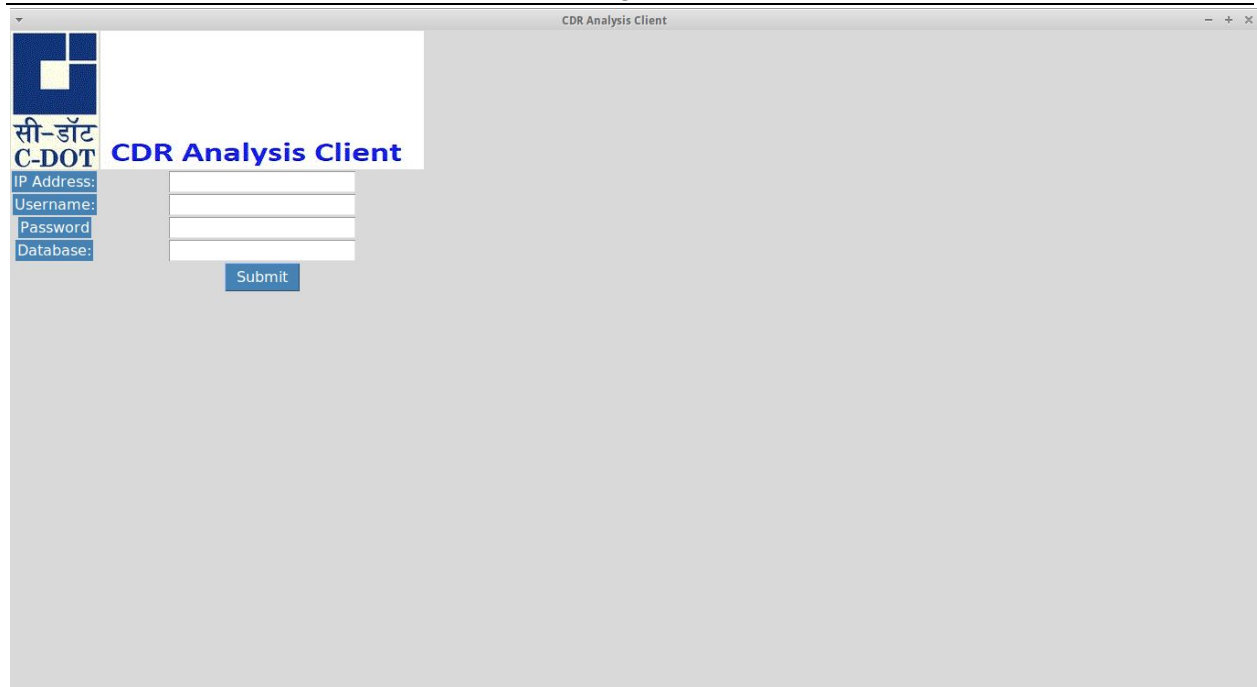


This screen provides a browsing window to select the file for uploading. CTRL button can be used to select multiple files.



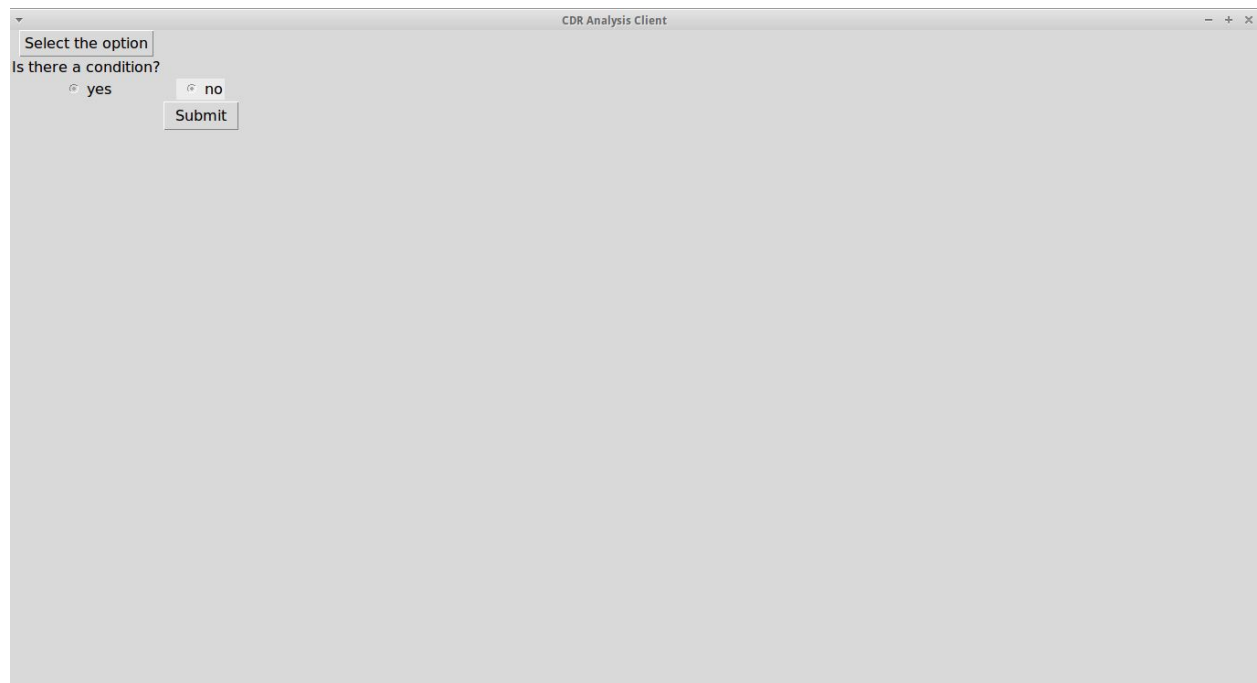
CLIENT :

The analyst has to log into the server



The screenshot shows a window titled "CDR Analysis Client". On the left, there is a logo with the text "सी-डॉट C-DOT" and a blue square icon. To the right of the logo, the text "CDR Analysis Client" is displayed in blue. Below the logo, there are four input fields labeled "IP Address:", "Username:", "Password:", and "Database:". To the right of these fields are four empty input boxes. Below the input boxes is a blue "Submit" button.

The Analyst can select the conditions and parameters.



The screenshot shows the same "CDR Analysis Client" window. A small dialog box is open in the top-left corner. The dialog box has a title bar "Select the option" and contains the text "Is there a condition?". Below this text are two radio buttons: "yes" and "no". The "no" radio button is selected. Below the radio buttons is a "Submit" button.

The Analyst can set conditions for the parameters.

The screenshot shows a window titled "CDR Analysis Client". Inside the window, there is a label "Select the variable" above a text input field. Below this, there is a label "Variable value(multiple values to be seperated by ,)" followed by another text input field. To the right of the second input field is a "Submit" button.

The data can be viewed from the screen.

The screenshot shows the same "CDR Analysis Client" window, but now it displays a list of data rows. Each row is preceded by a line of asterisks indicating the row number. The data for each row includes NGCPE id, Sequence Number, write_time stamp, caller no, and called no.

Row	NGCPE id	Sequence Number	write_time stamp	caller no	called no
1. row	BPL-PR	85755	00:00:21 01/04/2019	912762253452	9428224481
2. row	BPL-PR	85759	00:00:44 01/04/2019	918980223198	912762232064
3. row	BPL-PR	85758	00:00:59 01/04/2019	919664873987	912762222133
4. row	BPL-PR	85761	00:01:28 01/04/2019	9898636133	912762222133
5. row	BPL-PR	85764	00:01:47 01/04/2019	9737015539	912762221108
6. row	BPL-PR	85760	00:01:55 01/04/2019	8141886284	912762238189
7. row	BPL-PR	85763	00:02:05 01/04/2019	912762256570	21232444
8. row	BPL-PR	85765	00:02:47 01/04/2019	912762226546	9664490963
9. row	BPL-PR	85767			

3.1.2 Hardware Interfaces

Users can interact with the hardware components right from the beginning of the application. They can use the keyboard to enter their username, password and other credentials and navigate the interface and select options using a mouse.

The OS used during the development of the project is Ubuntu

A hard disk of a minimum of 100 GB holds the database and the software application.

A RAM of 16GB or above is preferred for the smooth functioning of the application.

Processor: Intel Core Xeon

Libraries used: MySQL Connector to connect the database with the software

3.1.3 Software Interfaces

Following are the prerequisites for this application

Operating System: - Ubuntu is chosen because of its ability to run Bash scripts

Database: MariaDB database to store and read the data aspect of the software

Pycharm 2019.1: - To implement the project.

Python- Java is chosen as the language for this software because of its user friendly nature and relevance to the implementation of the software functionalities.

.

4 Other Non-functional Requirements

4.1 Performance Requirements

1. The system server required 25 minutes to upload 37,58,879 call records into the database. Statically, the system should take 0.4ms to upload a single call record.
2. The time taken to view the default parameters for CDRs of 3 months in 210 seconds.
3. Fetching data and performing functions takes the usual compiler time like 35-45 seconds.

4.2 Software Quality Attributes

4.2.1 RELIABILITY

The software is completely reliable and all the data used throughout the functioning of the software is provided directly by the use.

4.2.2 MAINTAINABILITY

It will be easy to maintain every new version of the software due to the modular approach involved in its design and implementation.

4.2.3 USABILITY

Application will be user friendly, all the functionalities will be clearly defined .The interface will be easy to navigate, crisp and precise. No confusing elements shall be added. It will be easy for new users to learn about the system and for old users to access and modify data pertaining to them.

4.2.4 REUSABILITY

Since the software is carefully modularized and the functionalities are very clearly segregated among all the users of the software, it is easy to reuse the code.

4.2.5 INTEROPERABILITY

The system can easily work with other systems such as the database. The easy of access and modification of the database through the many functionalities of the software gives it the nature of interoperability and the potential for more systems to be added to collaborate with the software in the future.

Appendix A - Group Log

Meetings were conducted regularly between both team members since the project has been announced.

In the first week, the topic was chosen and all the relevant tools,databases and materials were discussed and gathered. Furthermore the general aim of the software was established.

In the second week, a general synopsis was agreed upon which was eventually finalised and tweaked for detail and submitted. The synopsis was expanded upon and the different functionalities were discussed.

After the submission of the synopsis, work on project elaboration began. All the different functionalities were specified in detail and a general idea of the software was established. The SRS and project making started subsequently. Team members discussed every aspect of the software and noted down the technicalities. Finally the use case, swimlane and the project srs were compiled.

SOURCE CODE-

SERVER-

```
from tkinter import *
from tkinter import filedialog
import os
import mysql.connector as mariadb

root = Tk()                                #opens parent window
root.withdraw()

def search_for_data_file():                # opens a browsing window to select data file
    lk=Toplevel()
    currdir = os.getcwd()
    tempdir = filedialog.askopenfilenames(parent=lk, initialdir=currdir, title='Please select file containing
data')
    filedata = list(tempdir)
    if len(filedata) > 0:
        print ("You chose: %s" % filedata)
        lk.destroy()
    return filedata

def search_for_format_file():              # opens a browsing window to select format file
    currdir = os.getcwd()
    tempdir = filedialog.askopenfilename(parent=root, initialdir=currdir, title='Please select file containing
data format')
    if len(tempdir) > 0:
        print ("You chose: %s" % tempdir)
    return tempdir

def Main():
    ff=search_for_format_file()
    file_format = open(ff, "r")
    mariadb_connection = mariadb.connect(user='root', password='root123', database='CDR')
    cursor = mariadb_connection.cursor()    # logging into MariaDB server
    num_lines = 0
    fstr = "
    ctbl = "
    with open(ff, 'r') as abc:
```

```

for line in abc:          # for counting number of lines in Format file
    num_lines += 1
for efg in range(num_lines):      # creating a string for CREATE TABLE command
    if efg == num_lines - 1:
        f = file_format.readline()
        fstr = fstr + f + ','a
        ctbl = ctbl + f + ' VARCHAR(40) ,'
    else:
        f = file_format.readline()
        f = f[:-1]
        fstr = fstr + f + ','
        ctbl = ctbl + f + ' VARCHAR(40) ,'
fstr = fstr + "Date_Inserted"
ctbl = ctbl + "Date_Inserted DATE"      # adding DATE INSERTED as a parameter in the table
fd = search_for_data_file()

y="CREATE TABLE IF NOT EXISTS cdr_data (" + ctbl + ");"
cursor.execute(y)          # running Maria DB command
for fdvar in fd:
    file_data = open(fdvar, "r")      # creating a string for INSERT INTO TABLE command
    j=0
    while 1:
        d = file_data.readline()
        if(j%2==0):                # only reading lines at the odd numbers in the data file
            for i in d:
                if(i=="\n"):
                    d=d[:-1]
            if d=="":
                file_data.close()
                break
            else:
                s=""
                for xyz in range(num_lines):
                    x = d.split(';')[xyz]
                    s = s + "" + x + "" + ','
                s = s + "DATE(current_timestamp)"
                s = "INSERT INTO cdr_data (" + fstr + ") VALUES (" + s + ");"
                cursor.execute(s)
            j += 1
        mariadb_connection.commit()    # save the database in its current state

if __name__ == "__main__":
    Main()

```

CLIENT-

```

import subprocess
import os
import tkinter
from tkinter import *

```

top = Tk()

```

def Query(i, j, k, l, p):    # getting the user input for the parameters defined in the Tkinter window
    ip=i.get()
    user=j.get()
    password=k.get()
    database=l.get()
    command_schema = "mysql" + " -h " + ip + " -u" + user + " -p" + password + " " + database + " -e
\"desc cdr_data;\""    # generating Bash script
    uj=subprocess.check_output(command_schema+"| awk 'NR>1{print $1}'" ,shell=True)
    uj=str(uj)    # running Bash script and storing the output as a string
    uj=uj[2:-3]
    col_names = uj.split("\n")    # getting parameter list

if col_names=="":    # a new window for invalid credentials
    uj=Toplevel()
    uj.geometry("1600x900")
    ko=Label(uj,text="CDR Analysis Client",font="42").grid(row=0,column=0)
    phk=PhotoImage(image=r"/root/Downloads/cdotlogo.gif")
    bn=Label(uj,image=phk).grid(row=0,column=1)
    io=Label(uj,text="Invalid Credentials",font="42").grid(row=1,column=0)
    bo=Button(uj,text='OK',font="42",command=uj.destroy).grid(row=2,column=0)
    uj.wait_window()
    top.destroy()
else:
    op = Toplevel()    # query window opens
    op.geometry("1600x900")
    mb = Menubutton(op, text="Select the option",font="42", relief=RAISED)
    mb.grid()
    mb.menu = Menu(mb, tearoff=0)
    mb["menu"] = mb.menu
    n = {}
    n["*"] = IntVar()
    n["Default "] = IntVar()
    for t in col_names:
        n[t] = IntVar()
    ty = []
    mb.menu.add_checkbutton(label="*(for all)",font="40",variable=n["*"])
    mb.menu.add_checkbutton(label="default", font="40", variable=n["Default "])
    for r in col_names:    # adding checkbutton for all and Default
        mb.menu.add_checkbutton(label=r, variable=n[r])
    mb.grid(row=1, column=0)    # adding checkbuttons for all parameters
    v=StringVar()
    kuy=Label(op,text="Is there a condition?",font="42").grid(row=2,column=0) #adding condition option
    r=Radiobutton(op,text="yes",font='42',variable=v,value="yes").grid(row=3,column=0)
    r1 = Radiobutton(op, text="no",font='42', variable=v, value="no").grid(row=3, column=1)
    bt=Button(op,text="Submit",font='42',command=op.destroy)
    bt.grid(row=4,column=1)
    op.wait_window()

if n["*"].get():
    ty.append("")
elif n["Default "].get():    # selecting five parameters for the default option

```

```

        ty.append("NGCPE_id,Sequence_Number,write_time_stamp,caller_no,called_no")
    else:
        for r in col_names:                # dynamic selection of parameters
            if nt[r].get():
                ty.append(r)

    cloumn = "
    for strvar in ty:
        cloumn = cloumn + strvar + ','
    cloumn = cloumn[:-1]                # generating a string for MARIA DB command
    condition= v.get()
    tyu=0
    if condition == 'yes':                # opens a new window for adding conditions
        tyu=1
        rt=Toplevel()
        mbt = Menubutton(rt, text="Select the variable",font="42", relief=RAISED)
        mbt.grid()
        mbt.menu = Menu(mbt, tearoff=0)
        mbt["menu"] = mbt.menu
        nt = {}
        for t in col_names:
            nt[t] = IntVar()
        tyt = []

        for r in col_names:
            mbt.menu.add_checkbutton(label=r, variable=nt[r])

        mbt.grid(row=1, column=0)

        v1=StringVar()
        ki=Label(rt,text="Variable value(multiple values to be seperated by
    ,)",font='42').grid(row=2,column=0)
        ei=Entry(rt,textvariable=v1).grid(row=2,column=1)
        bh=Button(rt,text="Submit",font='42',command=rt.destroy).grid(row=3,column=1)
        rt.wait_window()
        for r in col_names:
            if nt[r].get():
                tyt.append(r)
        va = ""
        va = v1.get()
        variable_value = va.split(',')
        fstr = "
        for (s1,s2) in zip(tyt,variable_value):    # generating Bash command
            fstr = fstr + s1 + '=' + s2 + ','
        fstr = fstr[:-1]
        command = "mysql" + " -h " + ip + " -u" + user + " -p" + password + " " + database + " -e \"select " + "
    " + cloumn + " from cdr_data where " + fstr + "\"G;\"/>/home/cdot/ffile.txt"
        else :
            command = "mysql"+" -h " + ip + " -u"+user+" -p"+password+" "+database+" -e \"select " + "
    "+cloumn+" from cdr_data\G;\"/>/home/cdot/ffile.txt"
        print(command)
        os.system(command)                # running Bash command
        ft=open("/home/cdot/ffile.txt",'r').read()    # opening file in which output is saved

```

```

lk=Toplevel()          # opening the Output window
lk.geometry("1600x900")
T=Text(lk,height=400,width=350)
scr = Scrollbar(lk)     # adding Scroll Bar

scr.pack(side=RIGHT, fill=Y)
scr.config(command=T.yview)
T.pack()
T.insert(END,ft)
bj=Button(lk,text="OK",font='42',command=lk.destroy).pack()    # adding Close button
scr = Scrollbar(lk)
T.configure(yscrollcommand=scr.set)
scr.pack(side=RIGHT, fill=Y)
lk.wait_window()
top.destroy()

def thuy():              # function for displaying Login window
    i = StringVar()
    j = StringVar()
    k = StringVar()
    l = StringVar()
    p = StringVar()
    top.title("CDR Analysis Client")
    top.geometry("1600x900")
    op=PhotoImage(file=r"/root/Downloads/CDOT_Logo.gif")
    opl=PhotoImage(file=r"/root/Downloads/CDRAC.gif")
    L=Label(image=op).grid(row=0,column=0)
    jk=Label(image=opl).grid(row=0,column=1)
    #ip=Label(top,text="CDR Analysis Client",font="120",foreground="steelblue").grid(row=0,column=1)
    L1 = Label(top, text="IP Address:",font="42",bg="steelblue",foreground="white")
    L1.grid(row=1, column=0)
    E1 = Entry(top, textvariable=i,font="42")
    E1.grid(row=1, column=1)
    L2 = Label(top, text="Username:",font="42",bg="steelblue",foreground="white")
    L2.grid(row=2, column=0)
    E2 = Entry(top, textvariable=j,font="42")
    E2.grid(row=2, column=1)
    L3 = Label(top, text="Password",font="42",bg="steelblue",foreground="white")
    L3.grid(row=3, column=0)
    E3 = Entry(top, textvariable=k,font="42")
    E3.grid(row=3, column=1)
    L4 = Label(top, text="Database:",font="42",bg="steelblue",foreground="white")
    L4.grid(row=4, column=0)
    E4 = Entry(top, textvariable=l,font="42")
    E4.grid(row=4, column=1)
    b = Button(top, text='Submit',font="42",bg="steelblue",foreground="white", command=lambda: Query(i,
j, k, l, p))
    b.grid(row=6, column=1)
    top.mainloop()

if __name__ == "__main__":
    thuy()

```

