# CHAPTER 1

## INTRODUCTION:

Bandwidth: The BW is a property of a medium. It is the difference between the highest and the lowest frequencies that the medium can satisfactorily pass.

Traffic: Transmission of messages through a communication network.

## SYMBOLS AND ABBREVATIONS

- BM: Bandwidth Monitor.
- TR: Traffic Reporter.
- BW: bandwidth

## DISADVANTAGES OF THE PRESENT SYSTEM:

1. The user will have no idea about the bandwidth usage and the traffic rates without having good software to identify the packets captured and find its traffic rates.

2. The user can’t keep track of the traffic rates.

3. Keeping the records of the bandwidth usage manually is impossible.

4. As huge data is to be maintained it consumes time and is tedious job.

## MOTIVATION

The main motivation to develop this software is that, present days usage of internet has increased rapidly and lot of upload and downloads activities have also increased. The maintenance of this upload and download rates is very important for the user so that he can know his bandwidth usage and can maintain it accordingly.

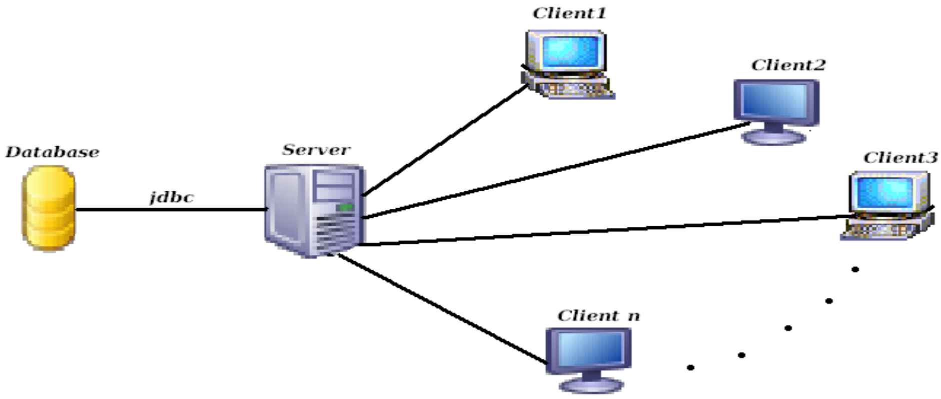
It also overcomes the disadvantages of the present system. The traffic rates can also be represented in the graphical form so that it becomes easy for the user to understand.

This project would be challenging as it would require stretching our knowledge of JAVA to its maximum limits, and also understanding the computer networks thoroughly in order to communicate between the client and server. And also understand the JDBC concept.

One more criteria that helped us to choose this project was the support of our guide without whose co-operation it would be difficult task

## PROBLEM STATEMENT

To develop software to Monitor the bandwidths of real time upload and download rates and generating traffic reports to user.



# CHAPTER 2

# SOFTWARE REQUIREMENT SPECIFICATION

## BANDWIDTH MONITOR & TRAFFIC REPORTER

## FUNCTIONAL & NON-FUNCTIONAL REQUIREMENTS:

Functional Requirement: Functional requirements are requirements which are the main components of the system that are necessary for the generation or specification of the system in accordingly. Like,

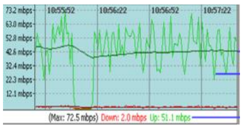
- Displays real time upload and download speeds in graphical and numerical form.
- Logs bandwidth usage.
- Providing daily, weekly and monthly bandwidth usage report.
- Bandwidth usage notification.

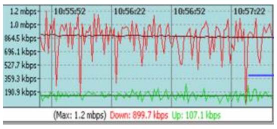
Non-Functional Requirement: Non-Functional requirements means as the name says non-functional which means, the components which are not as much necessary as functional, but they are used in the requirement phase of the system. Like,

- Storing and retrieving the traffic rates into database using JDBC
- Searching of daily, weekly and monthly traffic based on a given date, between 2 dates and based on IP address & one date , IP address and between given two dates.
- Providing graph for the traffic rates from the currently obtained traffic.

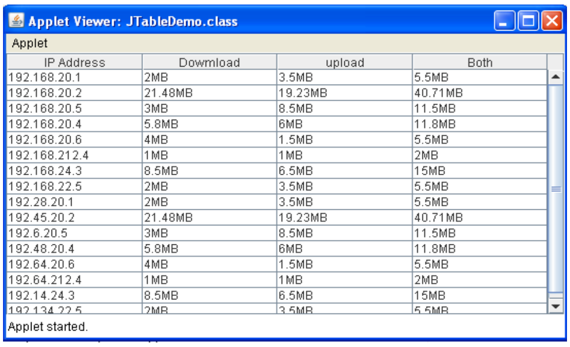
## REQUIREMENT OF BANDWIDTH MONITOR AND TRAFFIC REPORTER

- Assuming it to be simple client-server architecture.
- Set up the client server connection by giving the same port no to all the clients n the server and provide the server's IP address to all the clients.
- When the client gets connected to the server allow the client to start capturing the packets using JPCAP and by giving value 1. Then the client starts capturing and sends the captured packets length to the server
- The server accepts the data sent from the client and stores it into the database "traffic_report" in the table current_traffic.
- Graph-generates graph by fetching data from the current_traffic for the traffic rates i.e. upload, download and both.





- Traffic reporter-Based on one date : When the date is provided it looks in daily_traffic table if it finds it will be displayed.lly we have it for Based on 2 dates , Based on IP addresss and one date , Based on IP address and 2 dates.



| IP Address | Dowmload | upload | Both |
|---|---|---|---|
| 192.168.20.1 | 2MB | 3.5MB | 5.5MB |
| 192.168.20.2 | 21.48MB | 19.23MB | 40.71MB |
| 192.168.20.5 | 3MB | 8.5MB | 11.5MB |
| 192.168.20.4 | 5.8MB | 6MB | 11.8MB |
| 192.168.20.6 | 4MB | 1.5MB | 5.5MB |
| 192.168.212.4 | 1MB | 1MB | 2MB |
| 192.168.24.3 | 8.5MB | 6.5MB | 15MB |
| 192.168.22.5 | 2MB | 3.5MB | 5.5MB |
| 192.28.20.1 | 2MB | 3.5MB | 5.5MB |
| 192.45.20.2 | 21.48MB | 19.23MB | 40.71MB |
| 192.6.20.5 | 3MB | 8.5MB | 11.5MB |
| 192.48.20.4 | 5.8MB | 6MB | 11.8MB |
| 192.64.20.6 | 4MB | 1.5MB | 5.5MB |
| 192.64.212.4 | 1MB | 1MB | 2MB |
| 192.14.24.3 | 8.5MB | 6.5MB | 15MB |
| 192.134.22.5 | 2MB | 3.5MB | 5.5MB |

Applet started.

# CHAPTER 3

## HIGH LEVEL AND LOW LEVEL DESIGN:

### ARCHITECTURAL DESIGN:

- An architectural model represents an abstract view of the sub-systems making up the system.
- An architectural design may also include major information flows subsystems.
- It is usually represented as a block diagram.
- It may identify different types of functional components in the model.

### MODULE LEVEL REPRESENTATION WITH ALGORITHM:

- Module representation is the structural level where sub-systems are decomposed into modules.
- There are two types of modular representation, they are

  1. An object model where the system is decomposed into interacting objects.
  2. Another way is through representing it with algorithms for the particular Module.

Here in the Bandwidth Monitor & Traffic Report design presentation, the modules are divided into various categories,

- PacketMonitor
- BandwidthMonitor
- TrafficReporter

## PACKET MONITOR:

- This module helps us to Capture Packets & Send the Captured Packets length to the Server.
- Length is Upload and Download Rates.
- This module will work on the Client Machine.

## HIGH LEVEL DESIGN

### Classes:

- PacketMonitor
- SocketClient

**Algorithm for Packet Monitor:**

//Input: Usage of available interfaces for communication.

//Output: Captures the Packets during upload /Download.

Step1:Imports Jpcap package.

JPcap: is a Java class package which enables to capture and send IP packets from Java application. This package uses libpcap and Raw Socket API.

Using statements:

```
import jpcap.packet.*;
import jpcap.JpcapCaptor;
import jpcap.JpcapSender;
import jpcap.NetworkInterface;
import jpcap.NetworkInterfaceAddress;
```

Step1: Fetch available interfaces to listen on

Using inbuilt function getDevicelist().

Socket. These interfaces may be NIC, Ethernet card etc.

Step2:Based on Particular interface selected a device listener is setup.

```
Listens only to TCP/IP packets using setFilter() function.
Step 3:
 start listening for packets
while (true) {
Packet info = getPacket();
If packet content not null
 getPacketText(info) function called to retrieve the text in
 packet.
Listens separately for uploaded and downloaded data.
}
Step 4:getPacketText()
{//Input: Captured Packet
//Output: Length of Data in Packet.
return packet data in true text
Packet is taken and analysed.
byte[] bytes=new byte[pack.header.length + pack.data.length];
Extract the Header length of packet and copy it to variable Pack_header.
Extract the data length of packet using packet format and copy ito an variable
Pack_data.
StringBuffer buffer = new StringBuffer();
Create a String buffer for Storing the bytes of data.
Convert bytes in hexadecimal to binary.
 }
Step 5: Rate()
{
//Input: Length of Data in each packet
//Output: Uploaded or downloaded rate.
int download, upload;//initialised to 0
download<- download+Pack_data
Upload<-upload+pack_data
}
```

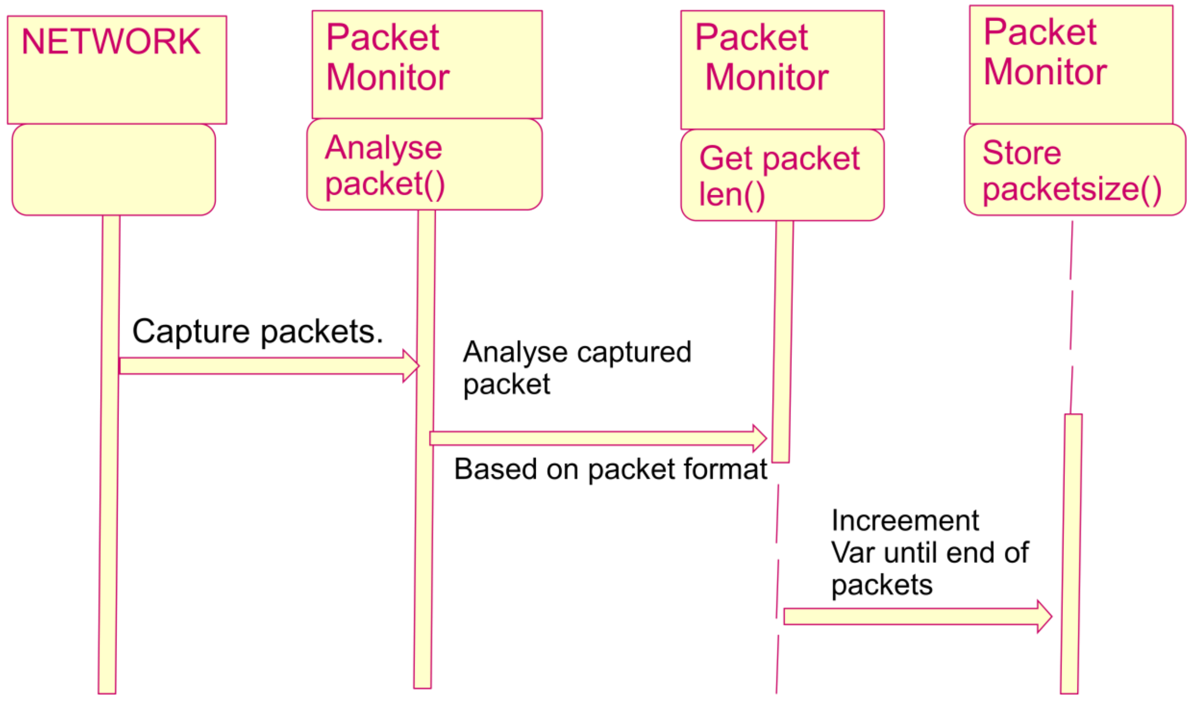
### Algorithm for Socket Client:

```
Step1:
   Socket socket = null;
   PrintWriter out = null;
   BufferedReader in = null;
Step2:
//Create socket connection
Void listenSocket()
{
 try{
Begin:
Create a new socket for client  specifying url as local host and with port address.
       socket = new Socket("localhost", 4450);
       create objects for printwriter and buffered reader classes.
     } catch Exceptions whereever mandatory.
 }
```

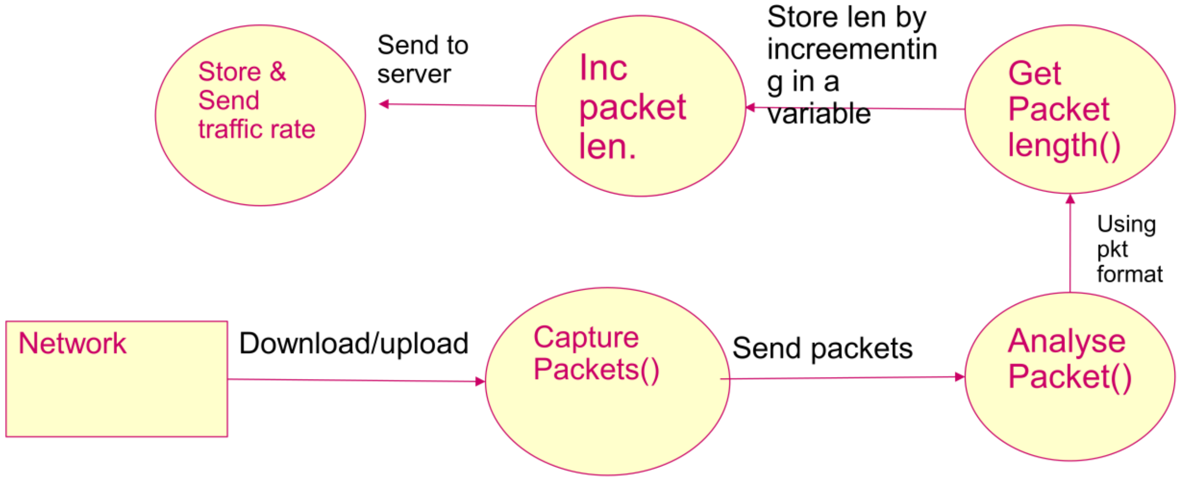
```
//Create socket connection
Void listenSocket()
{
 try{
Begin:
Create a new socket for client  specifying url as local host and with port address.
       socket = new Socket("localhost", 4450);
       create objects for printwriter and buffered reader classes.
     } catch Exceptions whereever mandatory.
 }
```

```
Void SendTraffic()
{
//Input: Uploaded and downloaded rate and IP address.
//Output: Server responds and stores the data in database.
Variables upload and download along with IP
 address of client machine sent to server after
 certain period.
}
```

## Sequence diagram for Packet Monitor:



## *PacketMonitor DFD :*



## BANDWIDTH MONITOR:

- This module helps us Receive Packets sent by Client.
- Store the Upload and Download Rates into the database.
- Generate Graph by fetching data from Current Traffic for Upload rates, Download rates and Total Rate.

## HIGH LEVEL DESIGN

### Classes:

- Bandwidth Monitor
- Graphtype

**Algorithm for Bandwidth Monitor:**

```
        //method to receive the traffic rates from the clients
    public void receiveTrafficRate(string IP,int upload,int download)

                    Step1: try{
                                server <-new ServerSocket(port no);
                            } catch (IOException e){
                              print the suitable statement for the exception
                                   }
                    Step2:try{
                              client <-server.accept();
                             } catch (IOException e) {
                         print the suitable statement for the exception
                                 }
                  Step3: try{
                        in <-new BufferedReader(new
                        InputStreamReader(client.getInputStream()));
                        out <- new PrintWriter(client.getOutputStream());
                        } catch (IOException e) {
                         print the suitable statement for the exception
                         }
//store the traffic rates sent from the client in database
      public void storeTrafficRate(string IP,int upload,int download)
                  {     //database  MYSQL Server
                        //Input: ip address, upload & download rates
                        Connection con <-null;
                        String url <-"jdbc:mysql://localhost:3306/";
                        String dbName <-"jdbctutorial";
                        String driverName <- "com.mysql.jdbc.Driver";
                        String userName <-"root";
                        String password <- "root";

                        Step1:try{
                               Class.forName(driverName).newInstance();
                              con <-DriverManager.getConnection(url+dbName,
                              userName, password);
                        Step2:try{
                                    Statement st <-con.createStatement();
```

```
            //if true enter the loop (i.e while loop)

            Step4: try{
                   line <- in.readLine();
           //Send data back to client
                  out.println(line);
                } catch (IOException e) {
                              print the suitable statement for the exception
                               }
            String table <- "CREATE TABLE  Current_traffic (IP_address
        varchar(15),uploadinMB  float,downloadinMB float)";
                         st.executeUpdate(table);
            String table <- "CREATE TABLE  Daliy_trafffic (Date date
              ,IP_address  varchar(15),uploadinMB float,downloadinMB float)";
                  st.executeUpdate(table);
                        }catch(SQLException s){
                        print suitable message for the exception
                        }//exception caught of inner try block
                  con.close();
                  }  catch (Exception e){
                  print suitable message for the exception
                  }//exception caught of outer try block
         }
```

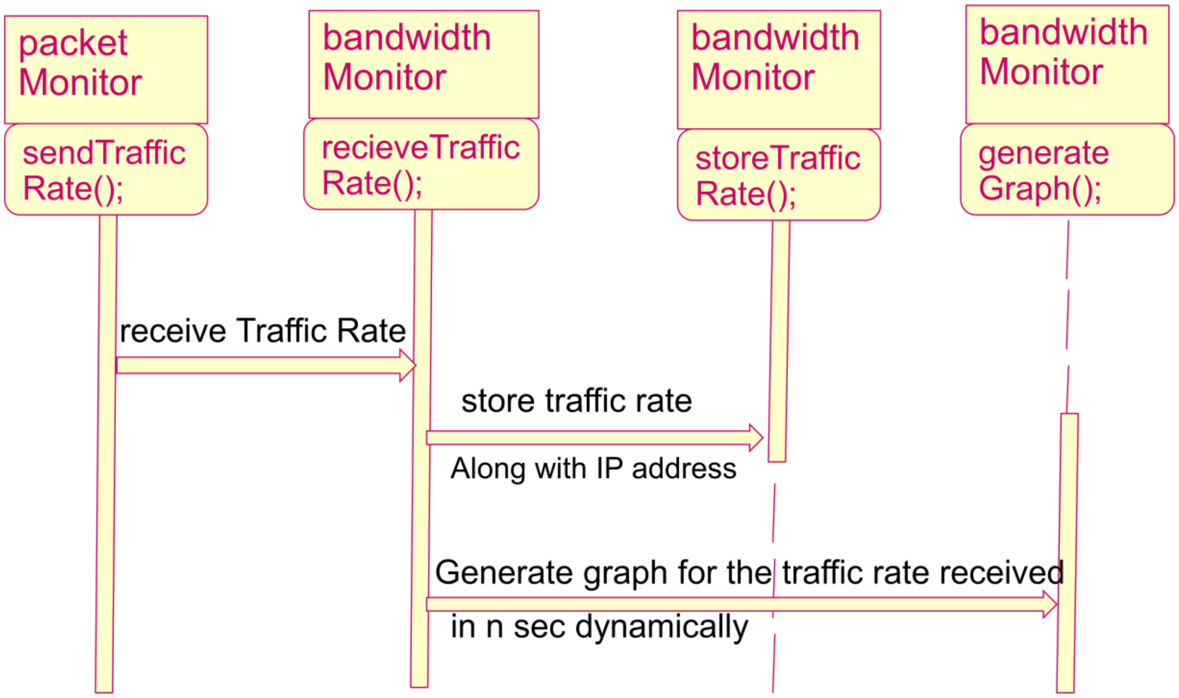
**Algorithm to Draw Graph:**

```
  Step1:
      Connection con = null;
      String url = "jdbc:mysql://localhost:3306/";
      String db = "traffic_report";
      String driver = "com.mysql.jdbc.Driver";
      String user = "root";
       String pass = "root";
 Step2:  try{
      Class.forName(driver).newInstance();
      con = DriverManager.getConnection(url+db, user, pass);
 Step3: try{
      Statement st = con.createStatement();
      ResultSet res = st.executeQuery("SELECT uploadinMB FROM
        current_traffic ");
       while (res.next()) {
       float i = res.getFloat("uploadinMB/downloadinMB/Total");
       data[j++]=i;
            }
    con.close();
   }
   catch (SQLException s){
    System.out.println("SQL code does not execute."+s);
   }
  }
catch (Exception e){
   e.printStackTrace();
  }
Step 4:      class GraphingData extends JPanel {

  protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D)g;
    g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
              RenderingHints.VALUE_ANTIALIAS_ON);
            getWidth();
            getHeight();
    // Draw ordinate.
    g2.draw(new Line2D.Double(PAD, PAD, PAD, h-PAD));
    // Draw abcissa.
    g2.draw(new Line2D.Double(PAD, h-PAD, w-PAD, h-PAD));
    // Draw labels.
    Font font = g2.getFont();
```

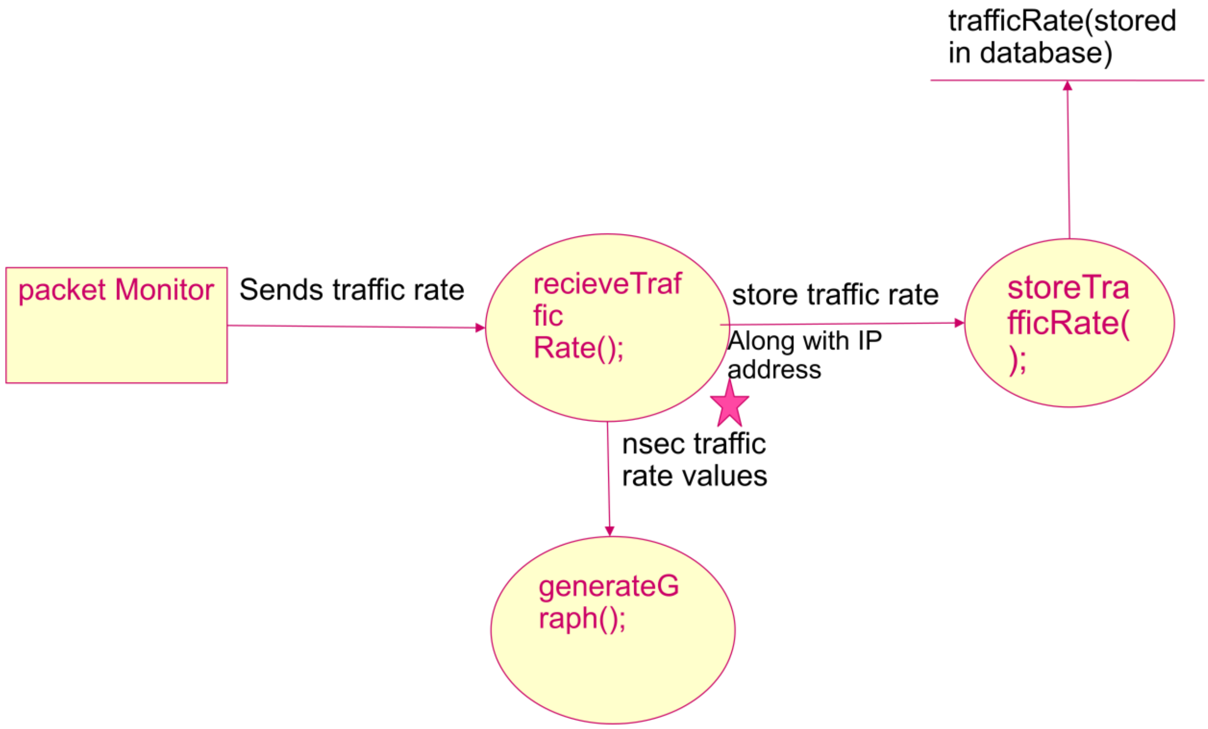
```
FontRenderContext frc = g2.getFontRenderContext();
LineMetrics lm = font.getLineMetrics("0", frc);
float sh = lm.getAscent() + lm.getDescent();
// Ordinate label.

float sy = PAD + ((h - 2*PAD) - s.length()*sh)/2 + lm.getAscent();
for(int i = 0; i < s.length(); i++) {
   String letter = String.valueOf(s.charAt(i));
   float sw = (float)font.getStringBounds(letter, frc).getWidth();
   float sx = (PAD - sw)/2;
   g2.drawString(letter, sx, sy);
   sy += sh;



// Abcissa label.
      g2.setPaint(Color.color.darker());
s = "TIME(in seconds)";
String pk="Initialize the X-axis values in Seconds";
sy = h - PAD + (PAD - sh)/2 + lm.getAscent();
float sw = (float)font.getStringBounds(s, frc).getWidth();
float sx = (w - sw)/2;
g2.drawString(s, sx, sy);
// Draw lines.
double xInc = (double)(w - 2*PAD)/(data.length-1);
double scale = (double)(h - 2*PAD)/getMax();
g2.setPaint(Color.color.darker());
for(int i = 0; i < data.length-1; i++) {
   double x1 = PAD + i*xInc;
   double y1 = h - PAD - scale*data[i];
   double x2 = PAD + (i+1)*xInc;
   double y2 = h - PAD - scale*data[i+1];
   g2.draw(new Line2D.Double(x1, y1, x2, y2));

   g2.drawString(pk, 20,h+1);
```

```
        }
        // Mark data points.
Step 5:       g2.setPaint(SET Color);
        for(int i = 0; i < data.length; i++) {
            double x = PAD + i*xInc;
            double y = h - PAD - scale*data[i];
            g2.fill(new Ellipse2D.Double(x-2, y-2, 4, 4));
        }
    }

    Step 6: private float getMax() {
            float max = -Integer.MAX_VALUE;
             for(int i = 0; i < data.length; i++) {
                        if(data[i] > max)
                         max = data[i];
                        }
                         return max;
                        }
                        }
 Step 7:   JFrame f = new JFrame();
       // f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.add(new GraphingData());
        f.setSize(width,height);
        f.setLocation(location);
        f.setVisible(true);
    }
```

## *Sequence diagram for Bandwidth Monitor:*



## *Bandwidth Monitor DFD :*



## TRAFIC REPORTER:

This module helps the user to know the Traffic Rates based on

- One date .
- Two dates (i.e b/w the given two dates).
- One date and IP_Address
- Two dates (i.e b/w the given two dates) and IP_Address

To Perform the above Operation data will be fetched from the database.

## HIGH LEVEL DESIGN

### Classes:

- getDailyReport()
- getReportBetween2Dates()

**Algorithm for Traffic Reporter:**

```
//Requires java.sql package
 getDailyReport()
//input   : Date.
//output : ResultSet object.
begin
        try
        begin
                load JDBC driver. //using forName() method
                establish connection to the backend database.//using getConnection() method.
                create Statement object.
                pass the query as parameter to executeUpdate().
                store the query result in ResultSet class object.
        end(try)
catch(Exception e)
        begin
                print stack trace using printStackTrace() method.
        end(catch)
```

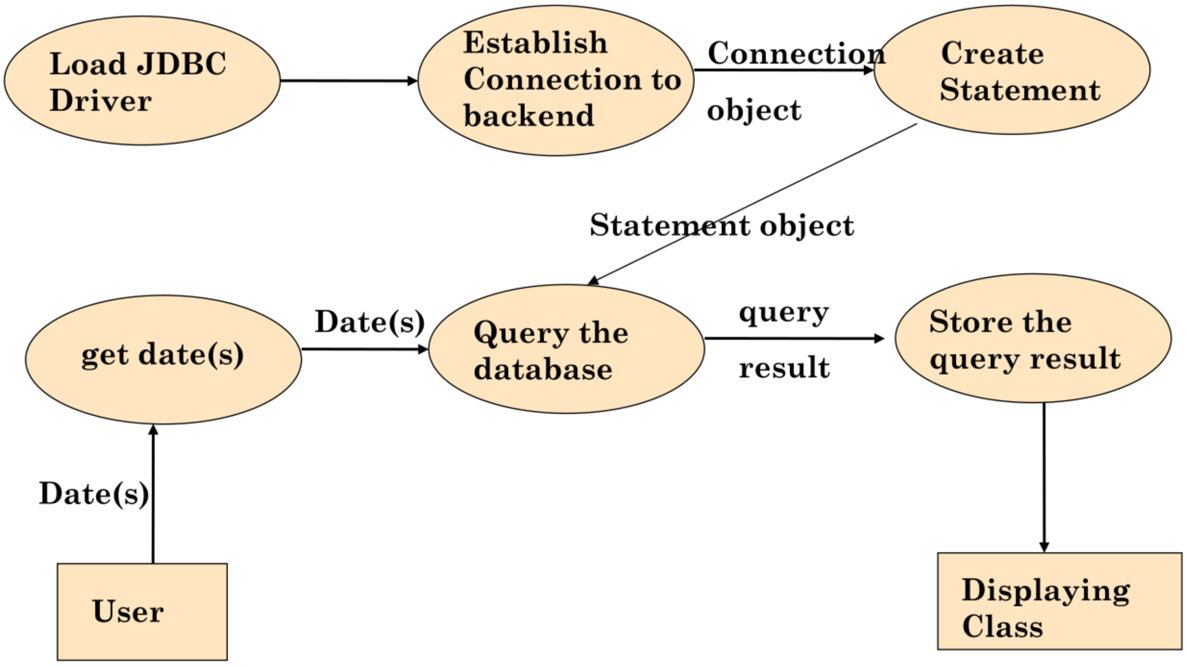
```
end (getDailyReport)

//Requires java.sql package
 getReportBetween2Dates()
//input   : From date, to Date.
//output : ResultSet object.
begin
        try
        begin
                load JDBC driver. //using forName() method
                establish connection to the backend database.//using getConnection() method.
                create Statement object.
                pass the query as parameter to executeUpdate().
                store the query result in ResultSet class object.
        end(try)
catch(Exception e)
        begin
                print stack trace using printStackTrace() method.
        end(catch)
end
```

**Algorithm for Display Daily Report :**

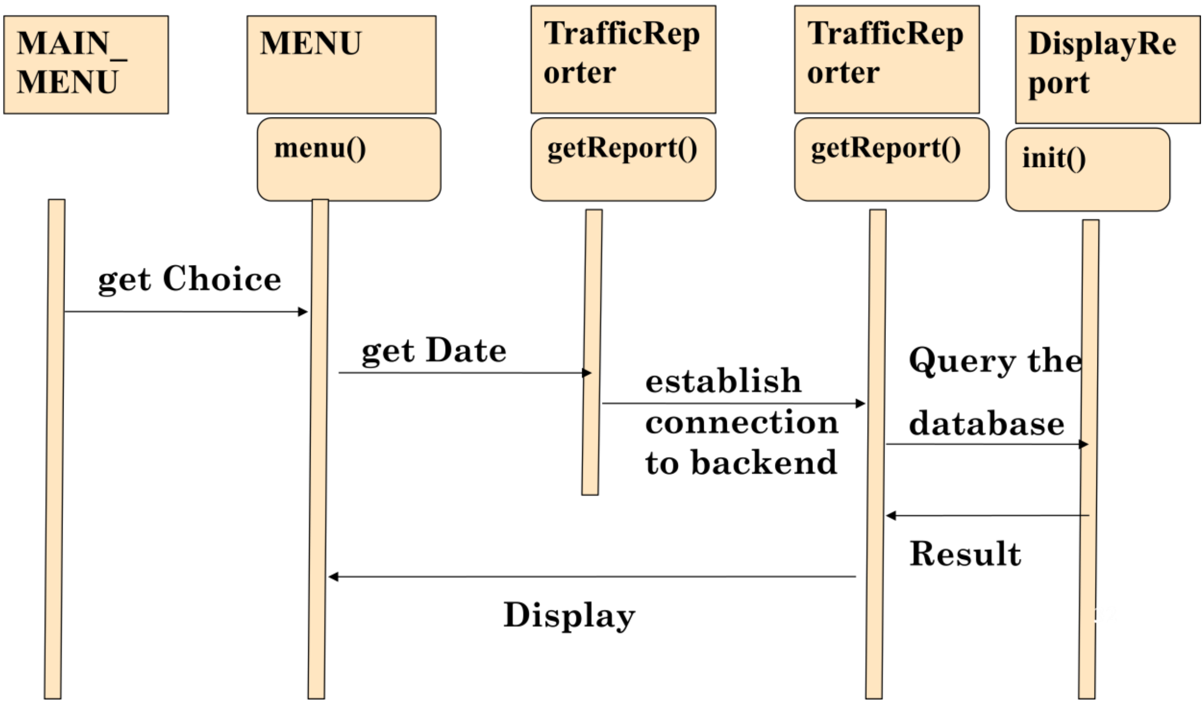
```
//Requires javax.swing and java.awt packages
//class containg this method need to extend from JApplet class
 init()
//input   :ResultSet object.
//output :displays Traffic of given date in tabular form.
does not return any thing.
begin
        get content pane and store it in Container class object.//use getContentPane() method.
        set layout manager //using setLayout(BorderLayout) method.
initialize column heading in 1D String array.
```

```
  initialize data in 2D Object array.
  create JTable object to create table & pass data,  heading created above as parameters.
  create JScrollPane object to include scroll bars.
       add it to content pane using add(JScrollPane) method.
end(init).
```

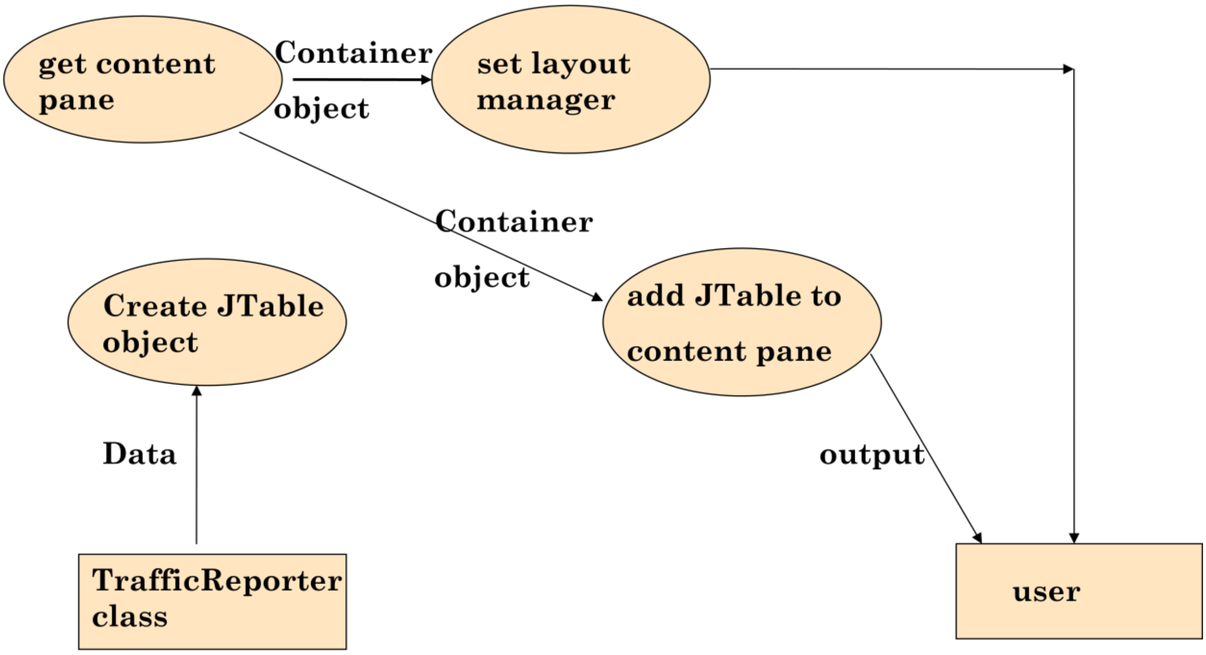
# DFD :



# Sequence Diagram :



# DFD:



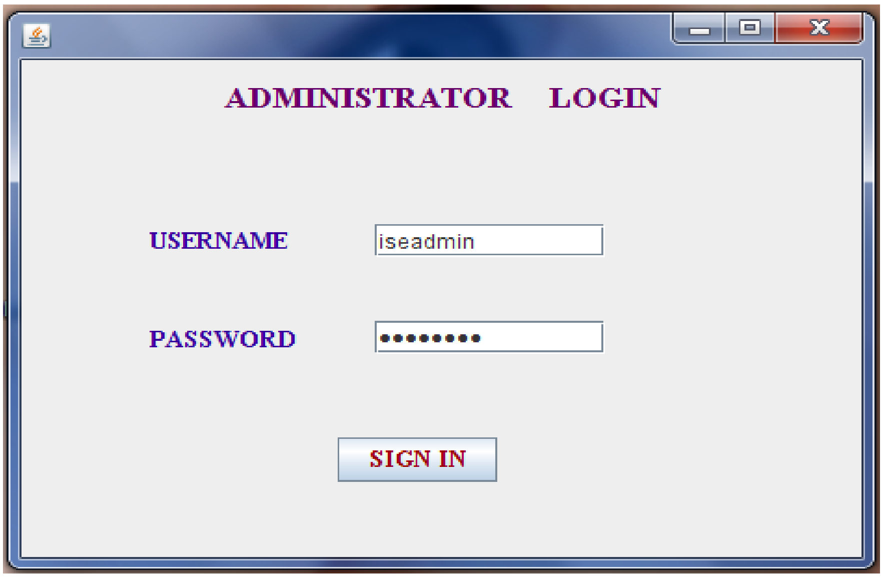
## Algorithm for displayBetween2Dates :

```
//Requires javax.swing and java.awt packages
//class containg this method need to extend from JApplet class
 init()
//input   :ResultSet object.
//output :displays Traffic of given 2 dates in tabular form.
does not return any thing.
begin
        get content pane and store it in Container class object.//use getContentPane() method.
        set layout manager //using setLayout(BorderLayout) method.

initialize column headings in 1D String array.
initialize data in 2D Object array.
create JTable object to create table & pass data, heading created above as parameters.
        create JScrollPane object to include scroll bars.
        add it to content pane using add(JScrollPane) method.
end(init).
```

# CHAPTER 4

## OUTPUTS

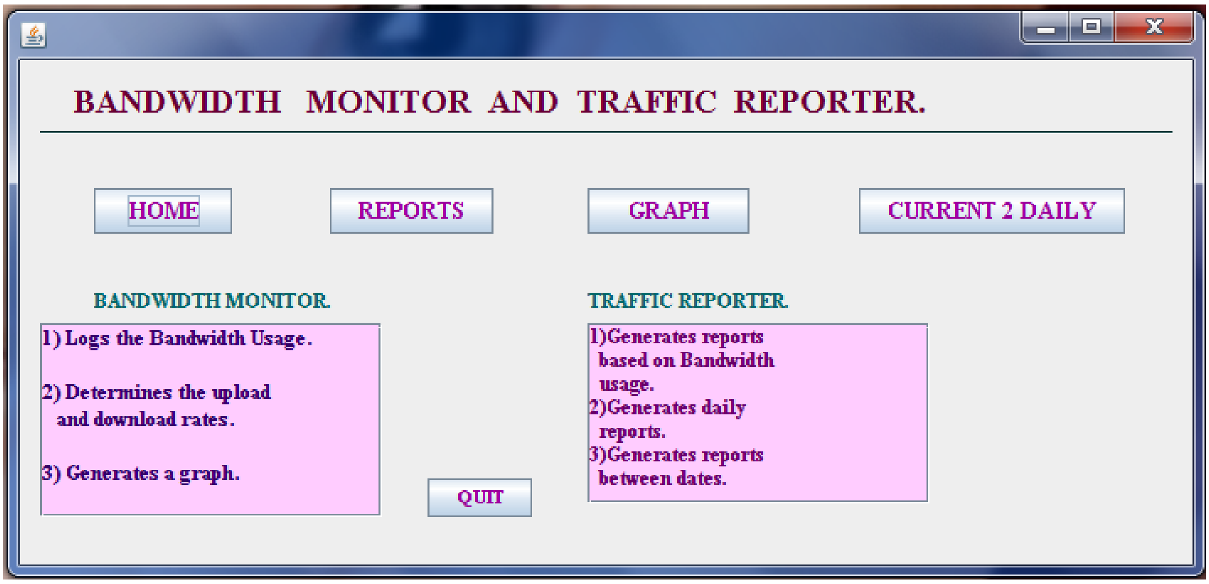
## SAMPLE SNAP SHOTS



ADMINISTRATOR LOGIN

USERNAME iseadmin

PASSWORD ●●●●●●●●●



BANDWIDTH MONITOR AND TRAFFIC REPORTER.

BANDWIDTH MONITOR.

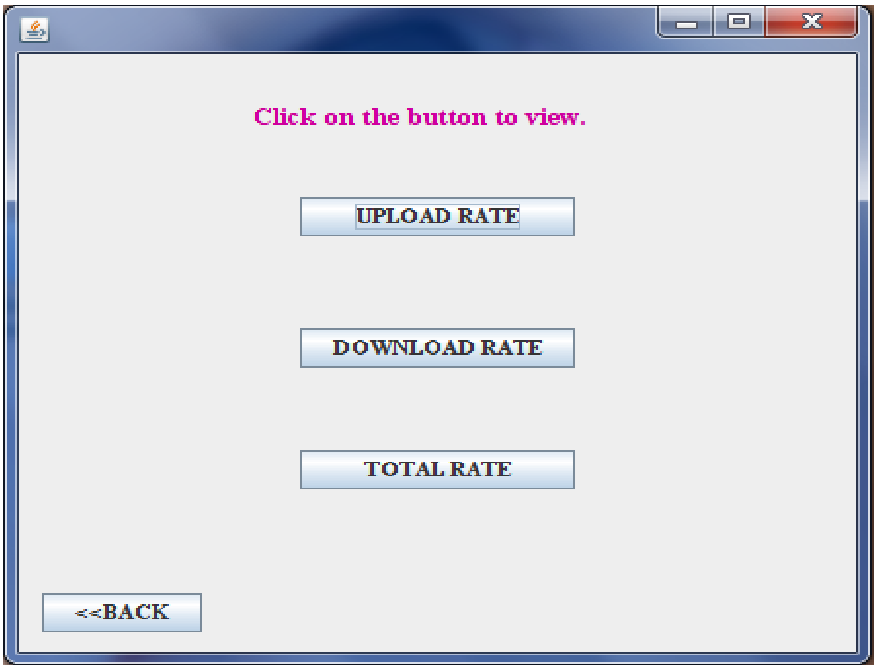
1) Logs the Bandwidth Usage.

2) Determines the upload and download rates.

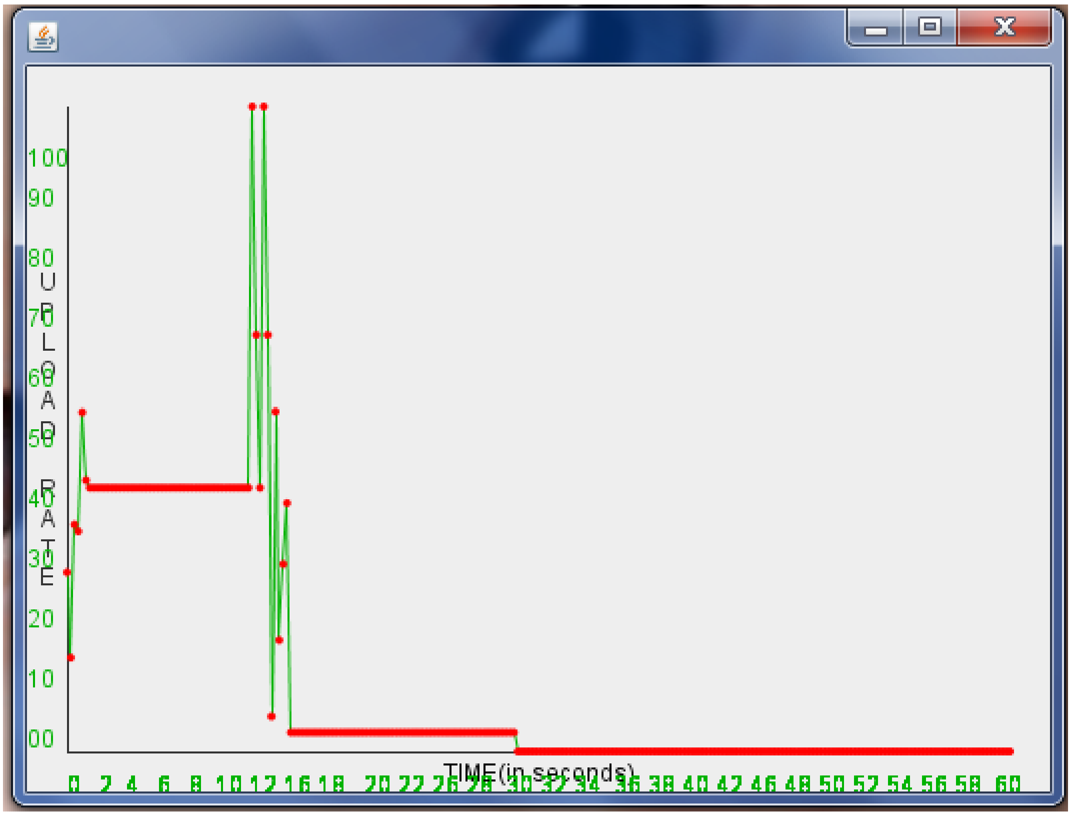
3) Generates a graph.

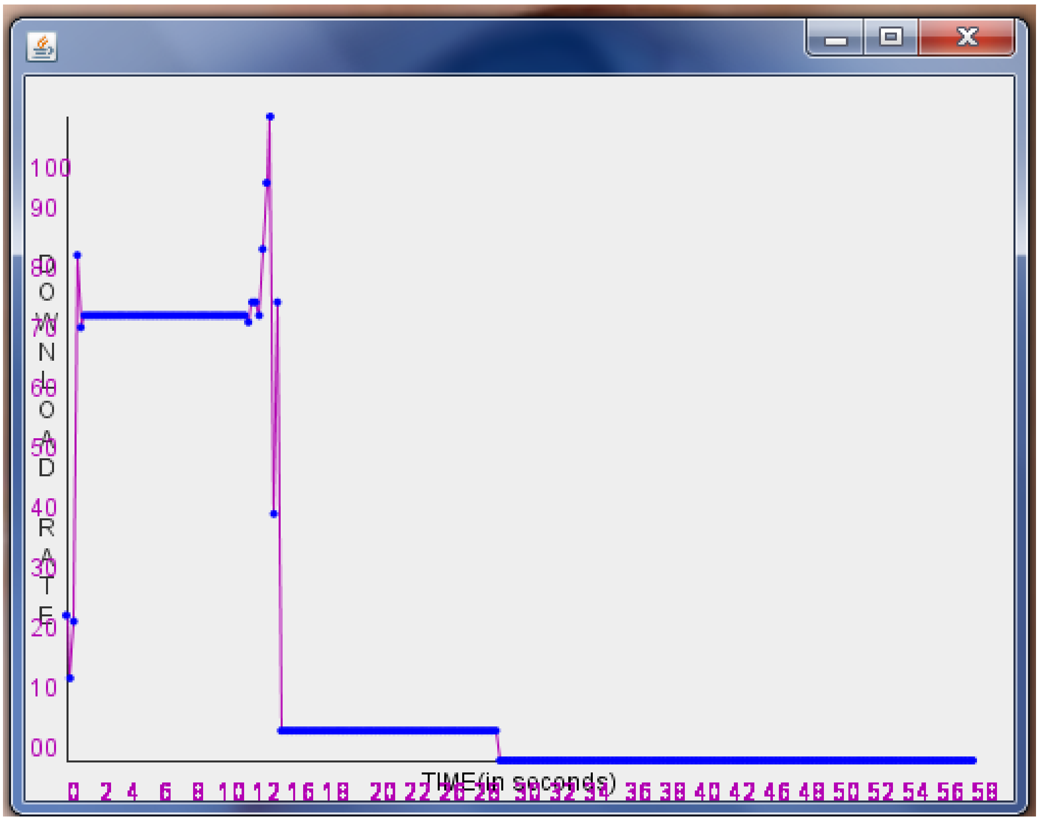
TRAFFIC REPORTER.

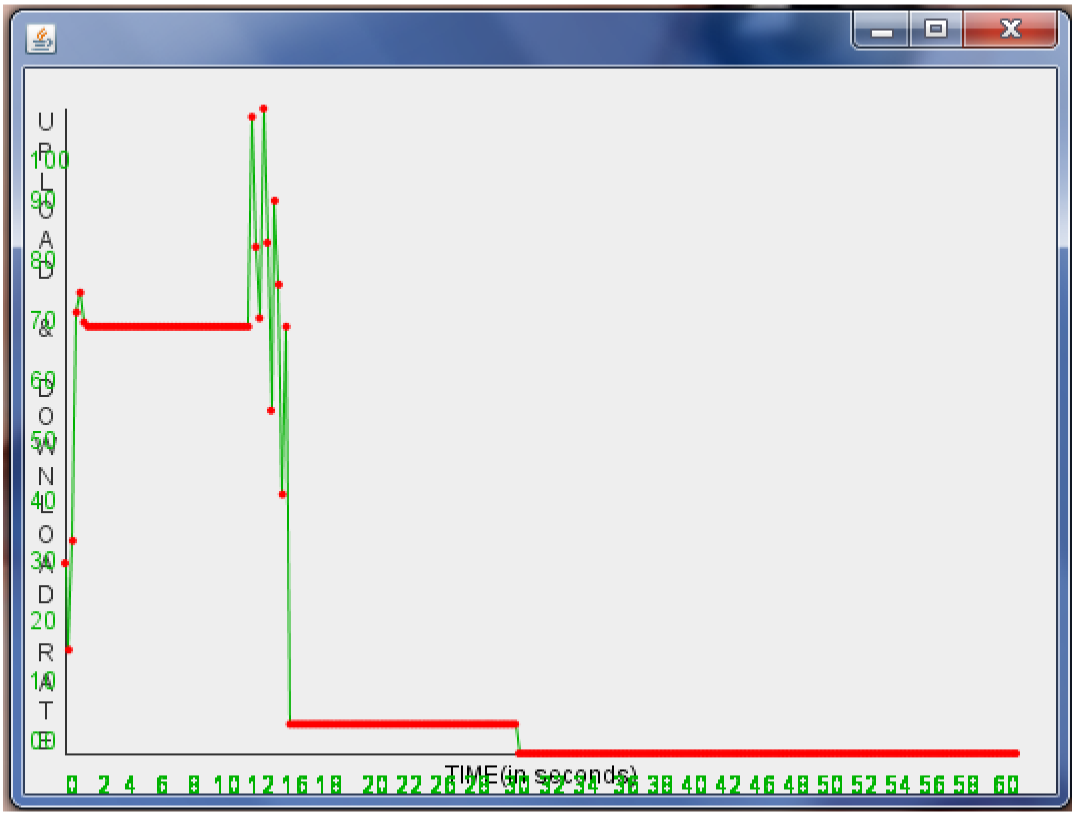
1)Generates reports based on Bandwidth usage.
2)Generates daily reports.
3)Generates reports between dates.

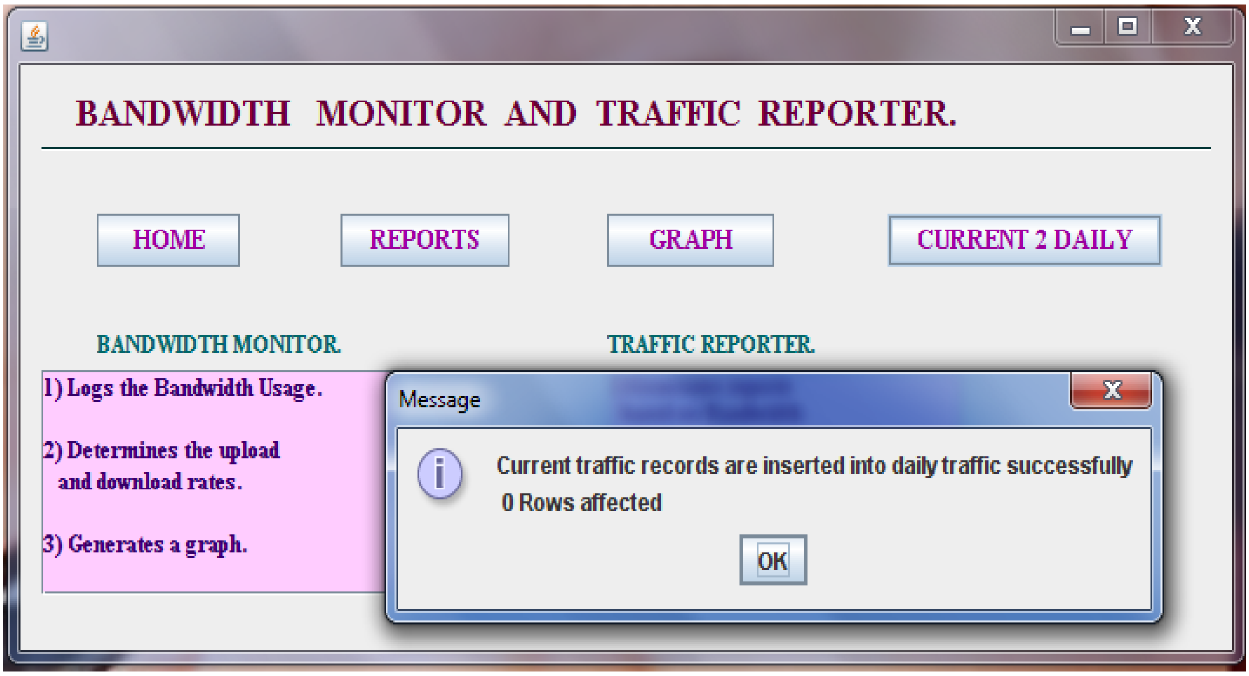


Click on the button to view.




100
90
80
70
60
50
40
30
20
10
00
D
O
W
N
L
O
A
D
R
A
T
E
TIME(in seconds)
0 2 4 6 8 10 12 16 18 20 22 26 28 30 32 34 36 38 40 42 46 48 50 52 54 56 58


100
90
80
70
60
50
40
30
20
10
00
U
P
L
O
A
D
&
D
O
W
N
L
O
A
D
R
A
T
E
TIME(in seconds)
0 2 4 6 8 10 12 16 18 20 22 26 28 30 32 34 36 38 40 42 46 48 50 52 54 56 58 60



BANDWIDTH MONITOR AND TRAFFIC REPORTER.

BANDWIDTH MONITOR.

TRAFFIC REPORTER.

1) Logs the Bandwidth Usage.

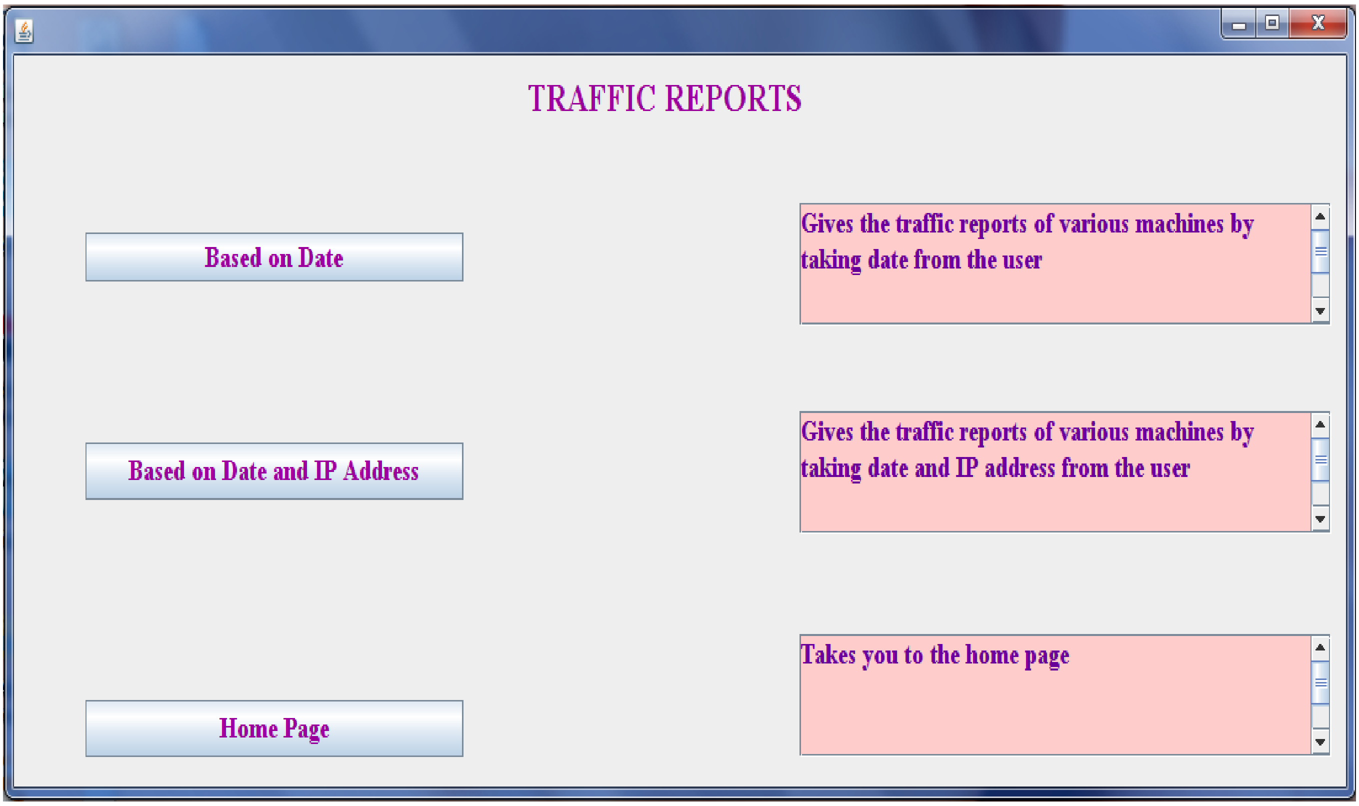
2) Determines the upload and download rates.

3) Generates a graph.

Message

Current traffic records are inserted into daily traffic successfully

0 Rows affected

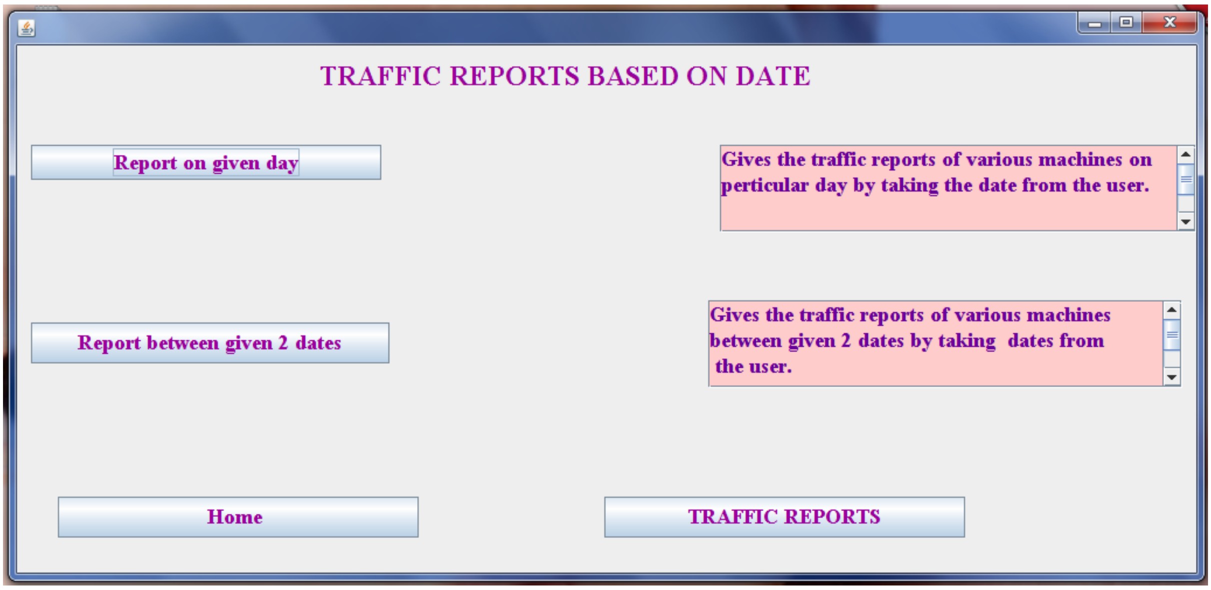


TRAFFIC REPORTS

Gives the traffic reports of various machines by taking date from the user

Gives the traffic reports of various machines by taking date and IP address from the user

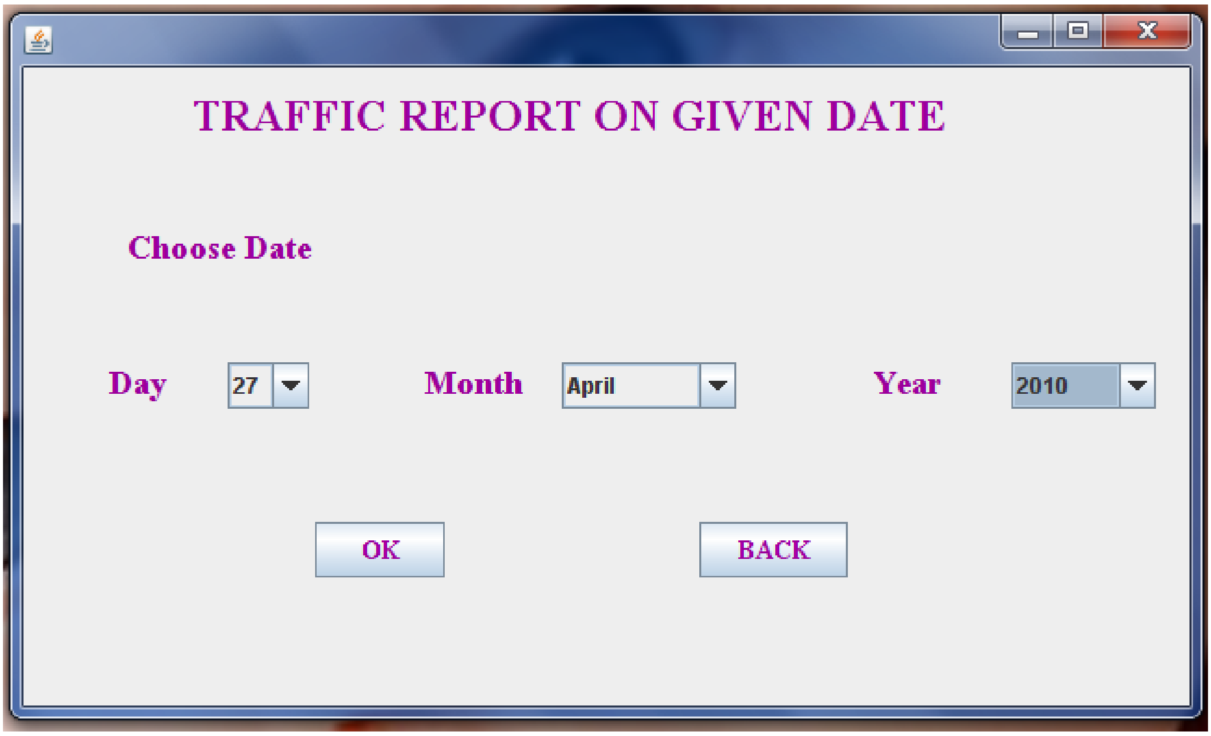
Takes you to the home page



TRAFFIC REPORTS BASED ON DATE

Gives the traffic reports of various machines on perticular day by taking the date from the user.

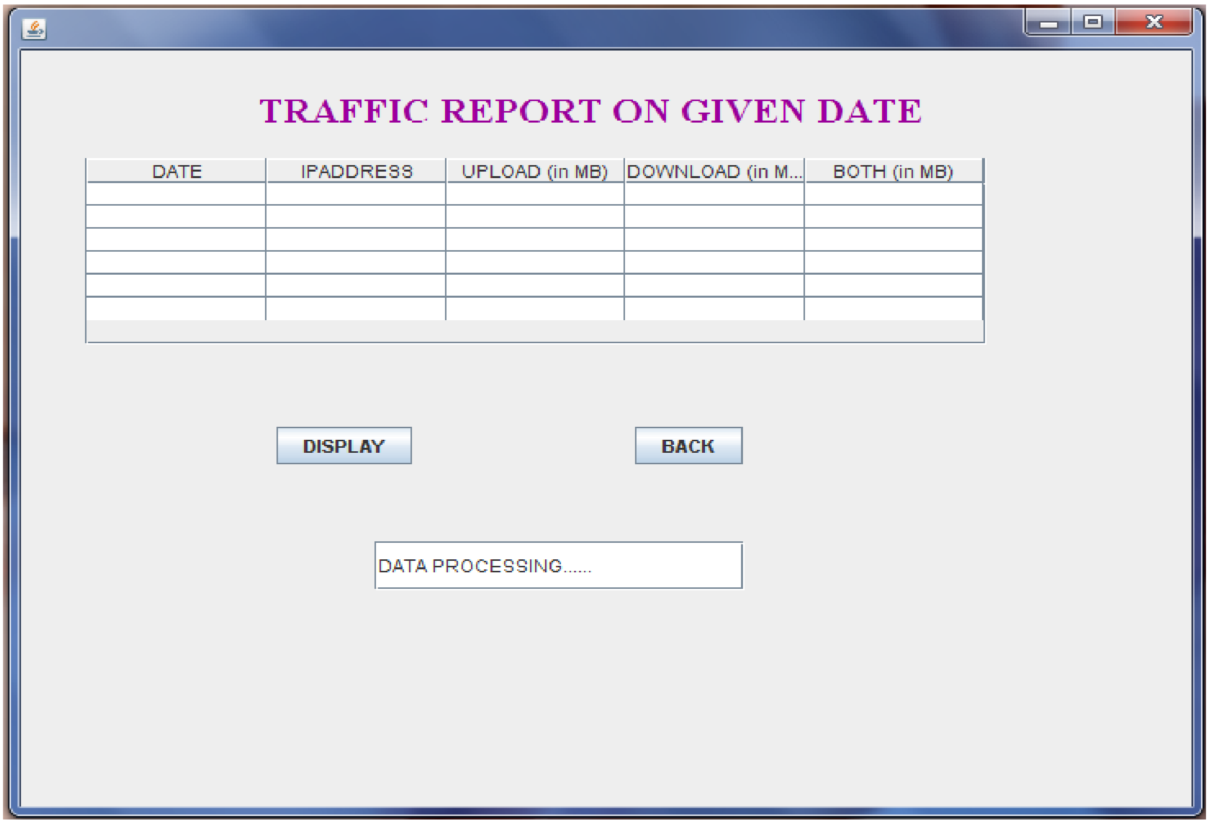
Gives the traffic reports of various machines between given 2 dates by taking dates from the user.



TRAFFIC REPORT ON GIVEN DATE

Choose Date

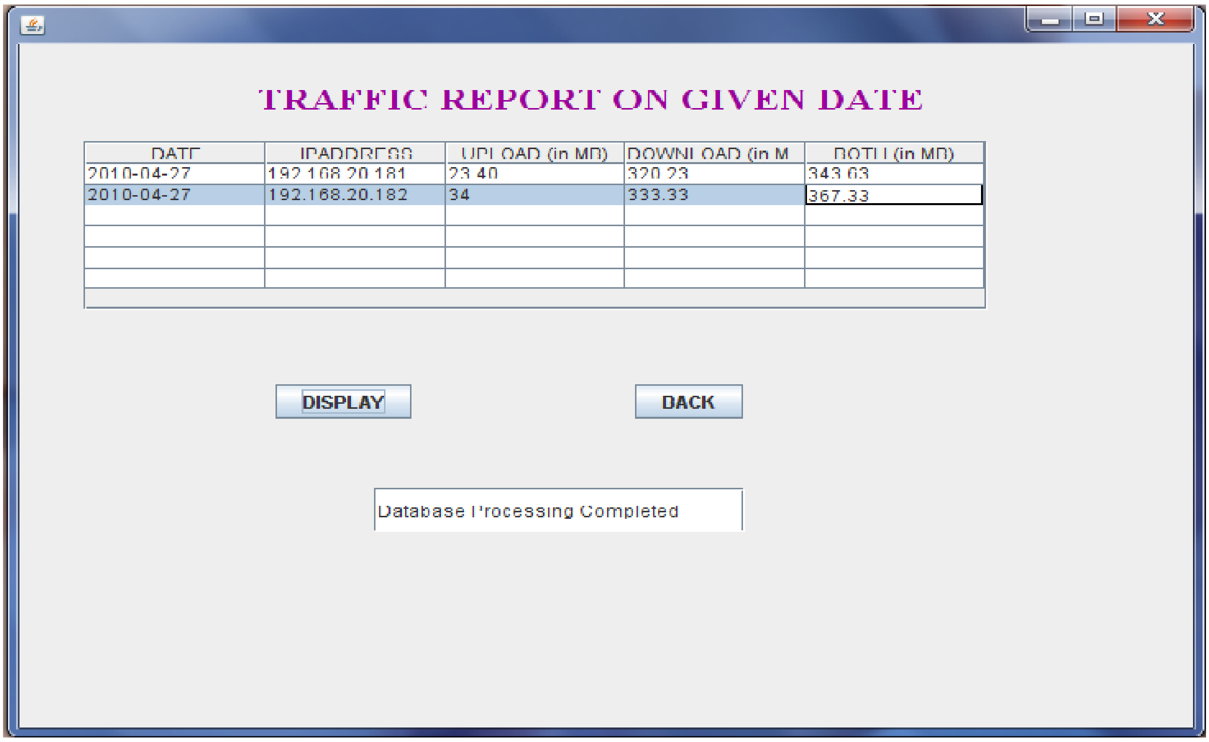
Day 27 Month April Year 2010



TRAFFIC REPORT ON GIVEN DATE

| DATE | IPADDRESS | UPLOAD (in MB) | DOWNLOAD (in M... | BOTH (in MB) |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

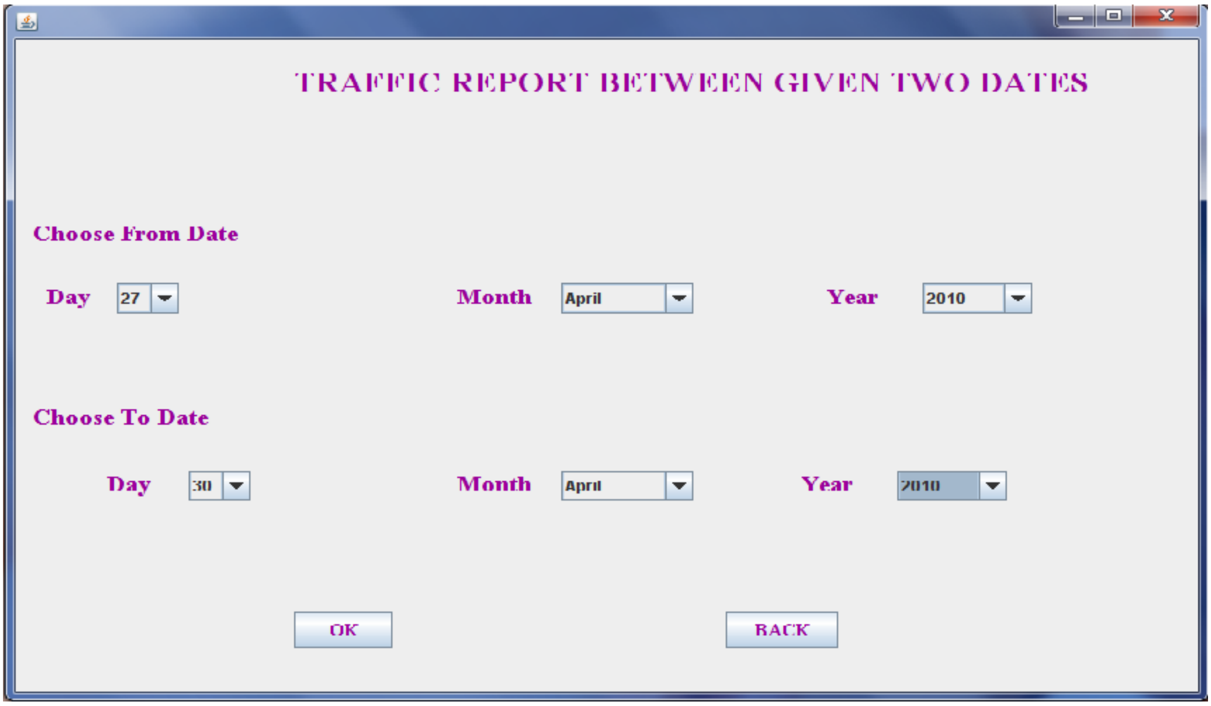
DATA PROCESSING......



TRAFFIC REPORT ON GIVEN DATE

| DATE | IPADDRESS | UPLOAD (in MB) | DOWNLOAD (in M | BOTH (in MB) |
|---|---|---|---|---|
| 2010-04-27 | 192.168.20.181 | 23.40 | 320.23 | 343.63 |
| 2010-04-27 | 192.168.20.182 | 34 | 333.33 | 367.33 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Database Processing Completed



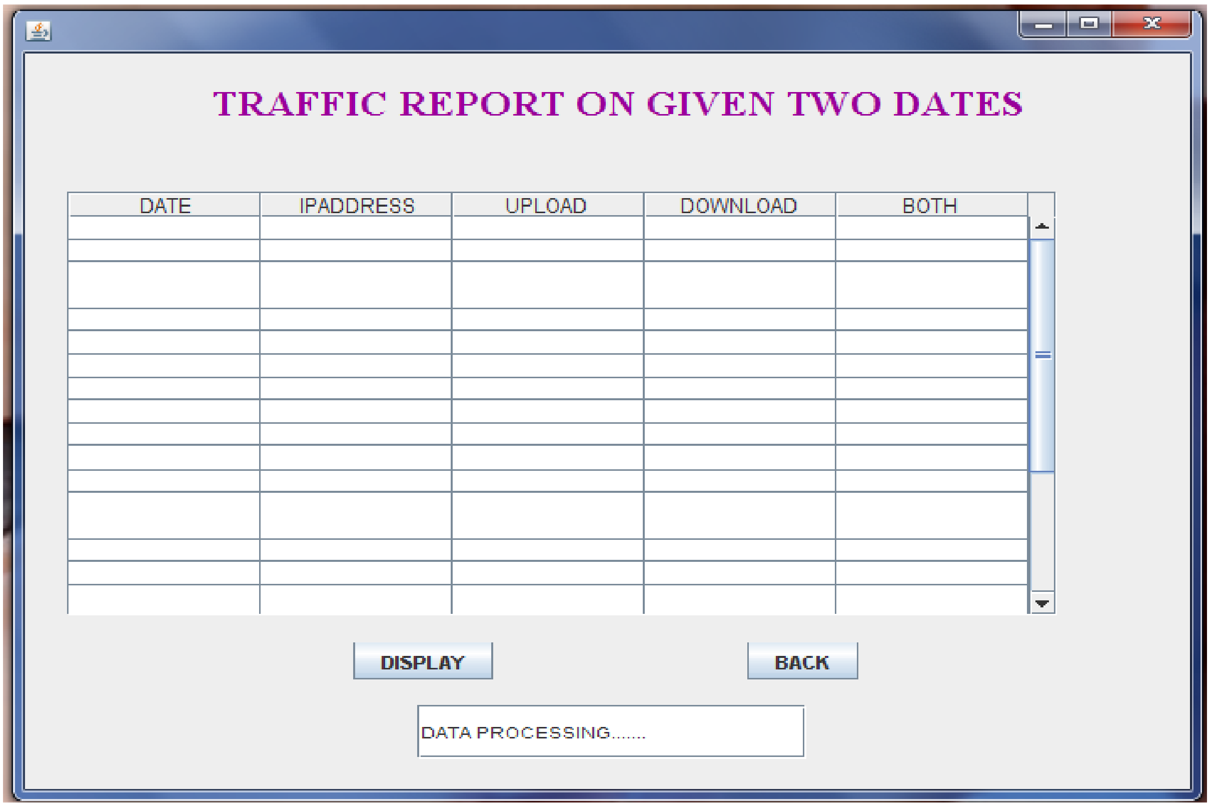
TRAFFIC REPORT BETWEEN GIVEN TWO DATES

Choose From Date

Day 27 Month April Year 2010

Choose To Date

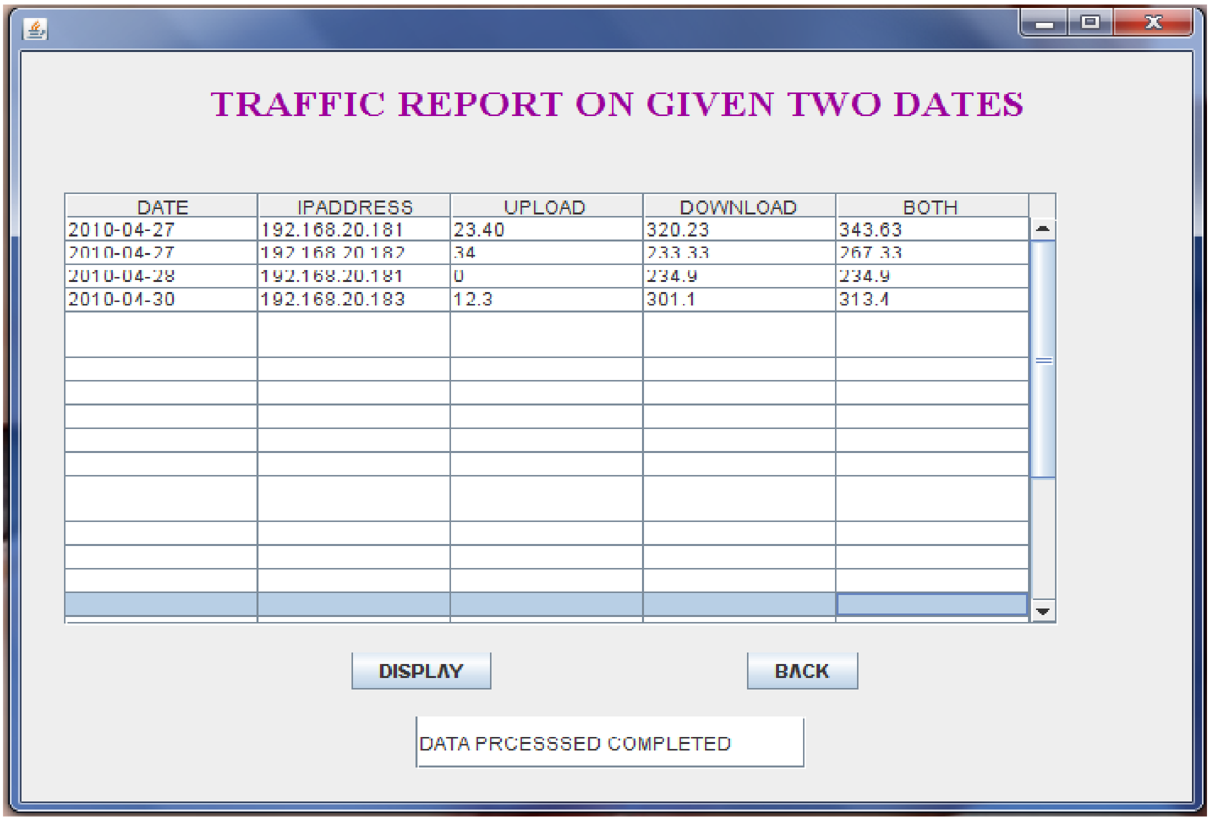
Day 30 Month April Year 2010



TRAFFIC REPORT ON GIVEN TWO DATES

| DATE | IPADDRESS | UPLOAD | DOWNLOAD | BOTH |
|---|---|---|---|---|

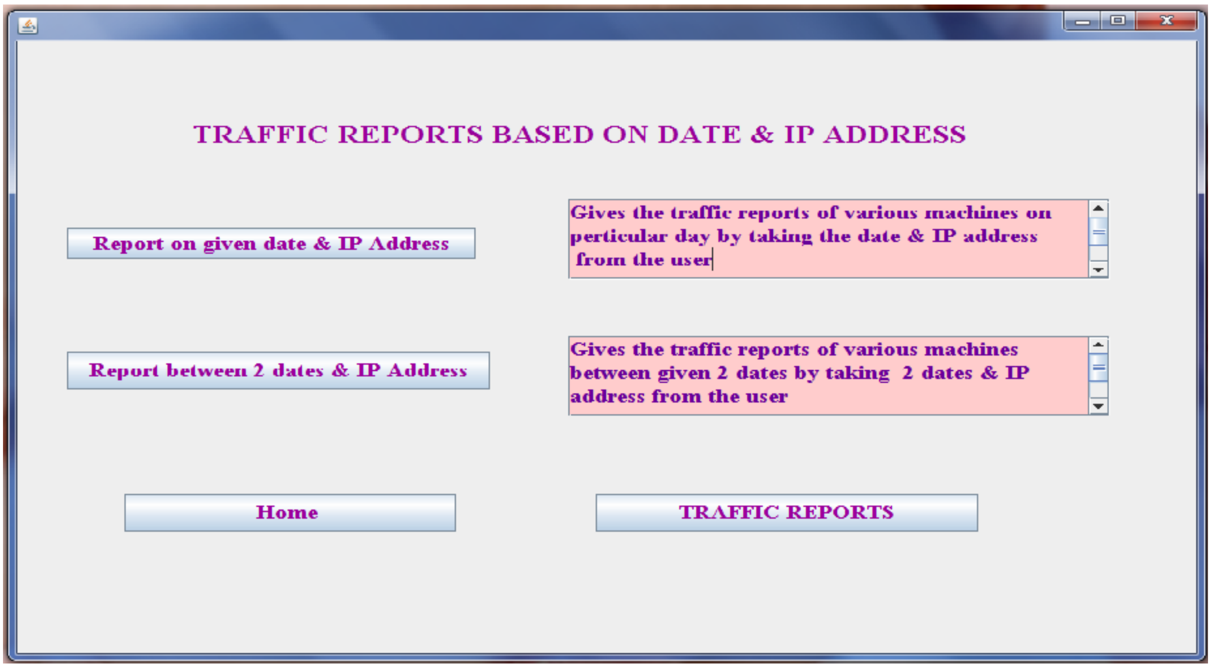
DATA PROCESSING.......



TRAFFIC REPORT ON GIVEN TWO DATES

| DATE | IPADDRESS | UPLOAD | DOWNLOAD | BOTH |
|---|---|---|---|---|
| 2010-04-27 | 192.168.20.181 | 23.40 | 320.23 | 343.63 |
| 2010-04-27 | 192.168.20.182 | 34 | 233.33 | 267.33 |
| 2010-04-28 | 192.168.20.181 | 0 | 234.9 | 234.9 |
| 2010-04-30 | 192.168.20.183 | 12.3 | 301.1 | 313.4 |

DATA PRCESSSED COMPLETED

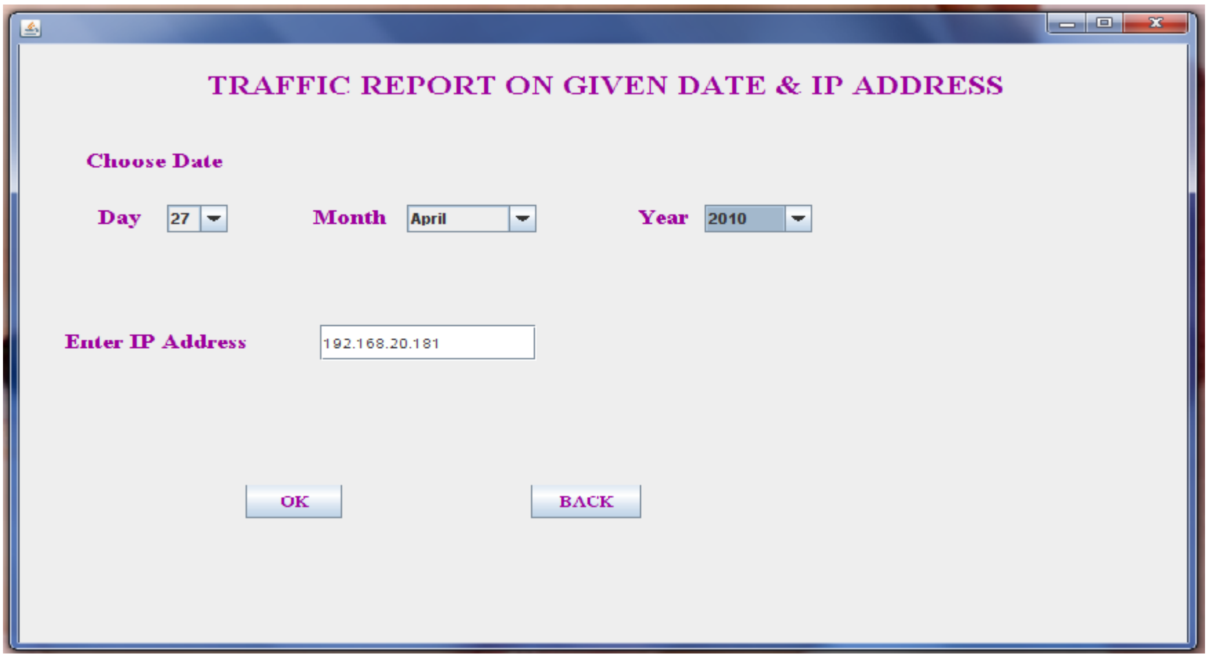


TRAFFIC REPORTS BASED ON DATE & IP ADDRESS

Gives the traffic reports of various machines on perticular day by taking the date & IP address from the user

Gives the traffic reports of various machines between given 2 dates by taking 2 dates & IP address from the user

Report on given date & IP Address
Report between 2 dates & IP Address
Home
TRAFFIC REPORTS

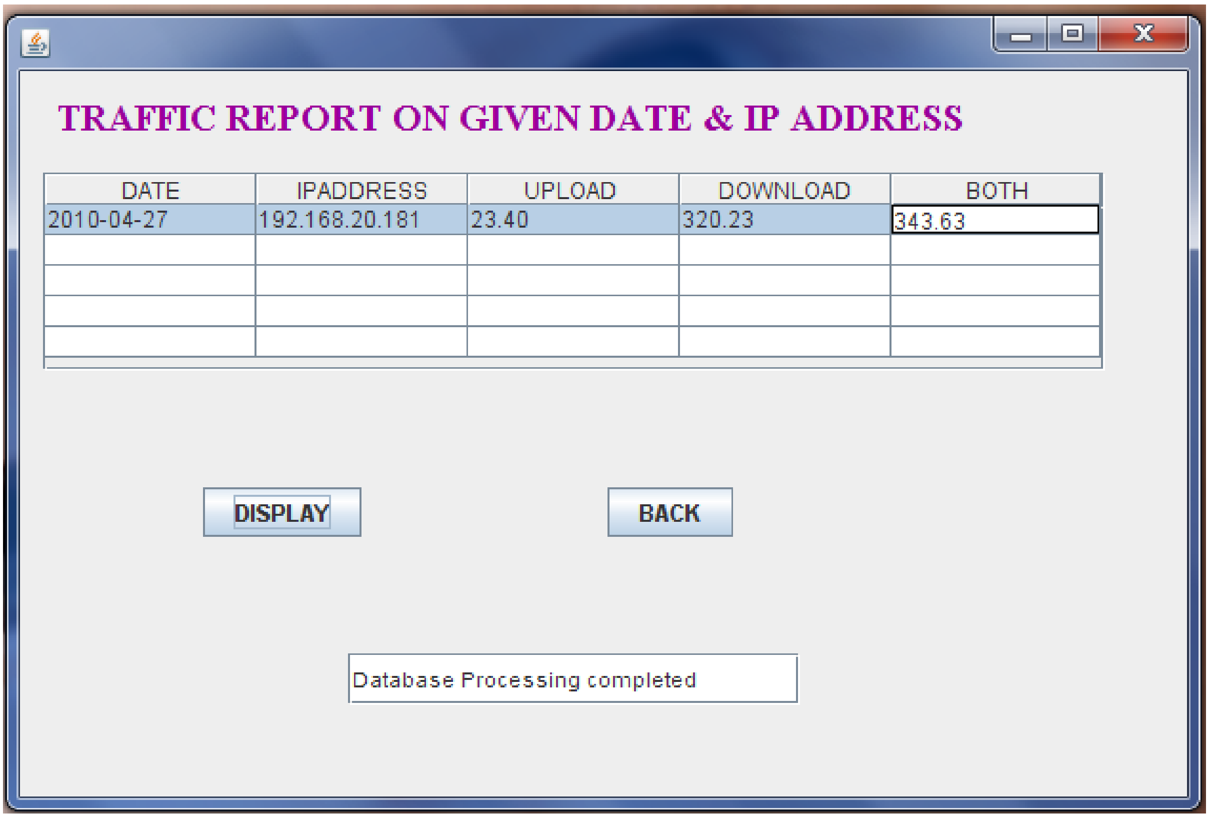


TRAFFIC REPORT ON GIVEN DATE & IP ADDRESS

Choose Date

Day 27 Month April Year 2010

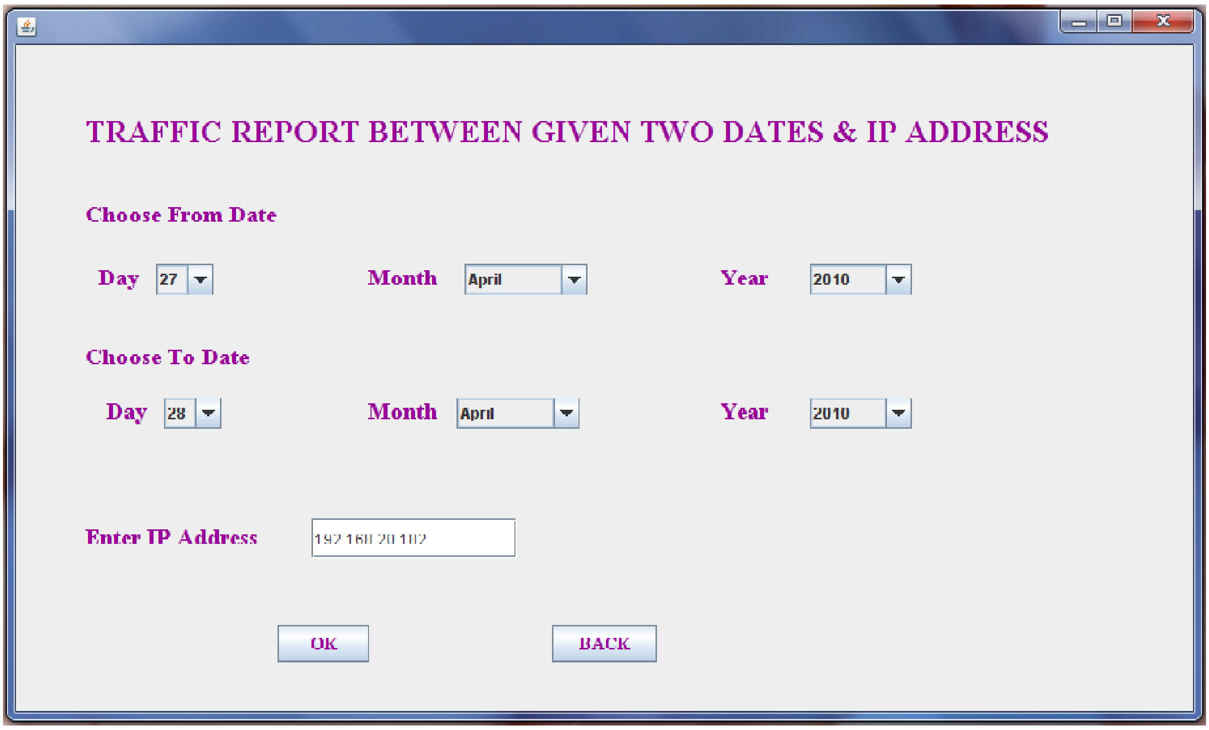
Enter IP Address 192.168.20.181



TRAFFIC REPORT ON GIVEN DATE & IP ADDRESS

| DATE | IPADDRESS | UPLOAD | DOWNLOAD | BOTH |
|---|---|---|---|---|
| 2010-04-27 | 192.168.20.181 | 23.40 | 320.23 | 343.63 |

Database Processing completed



TRAFFIC REPORT BETWEEN GIVEN TWO DATES & IP ADDRESS

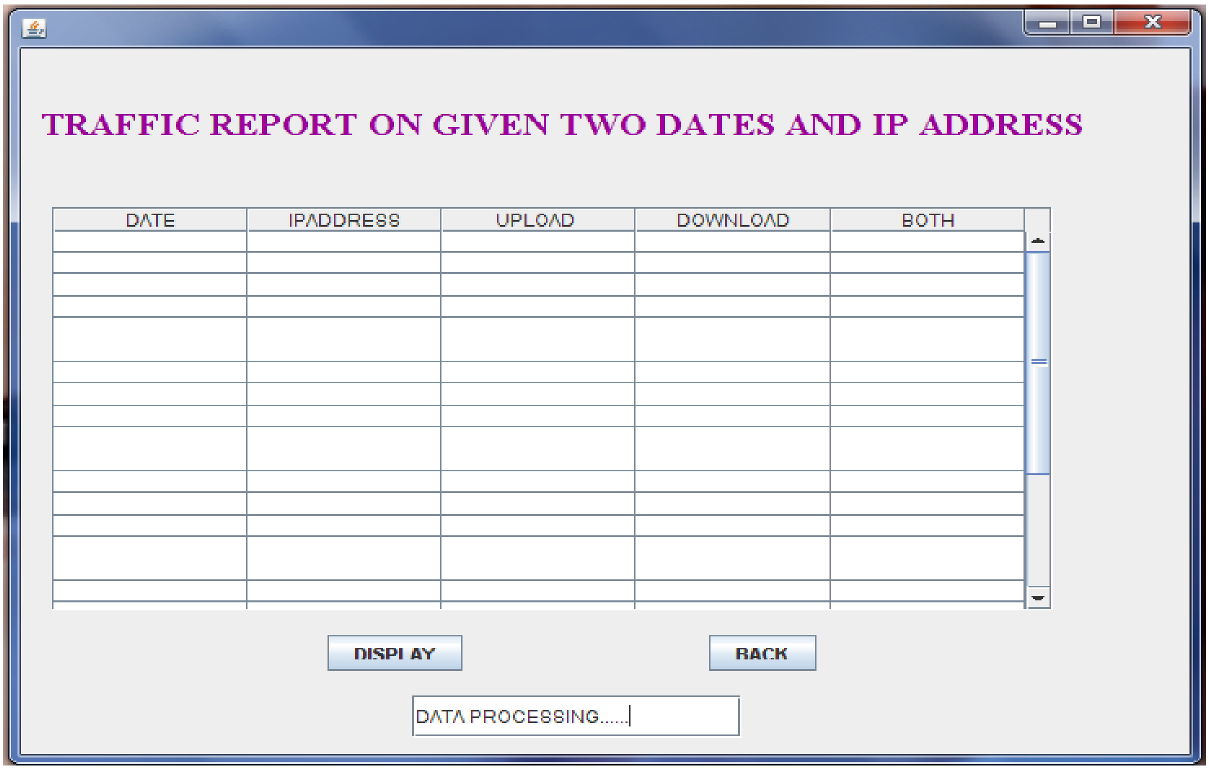
Choose From Date

Day 27 Month April Year 2010

Choose To Date

Day 28 Month April Year 2010

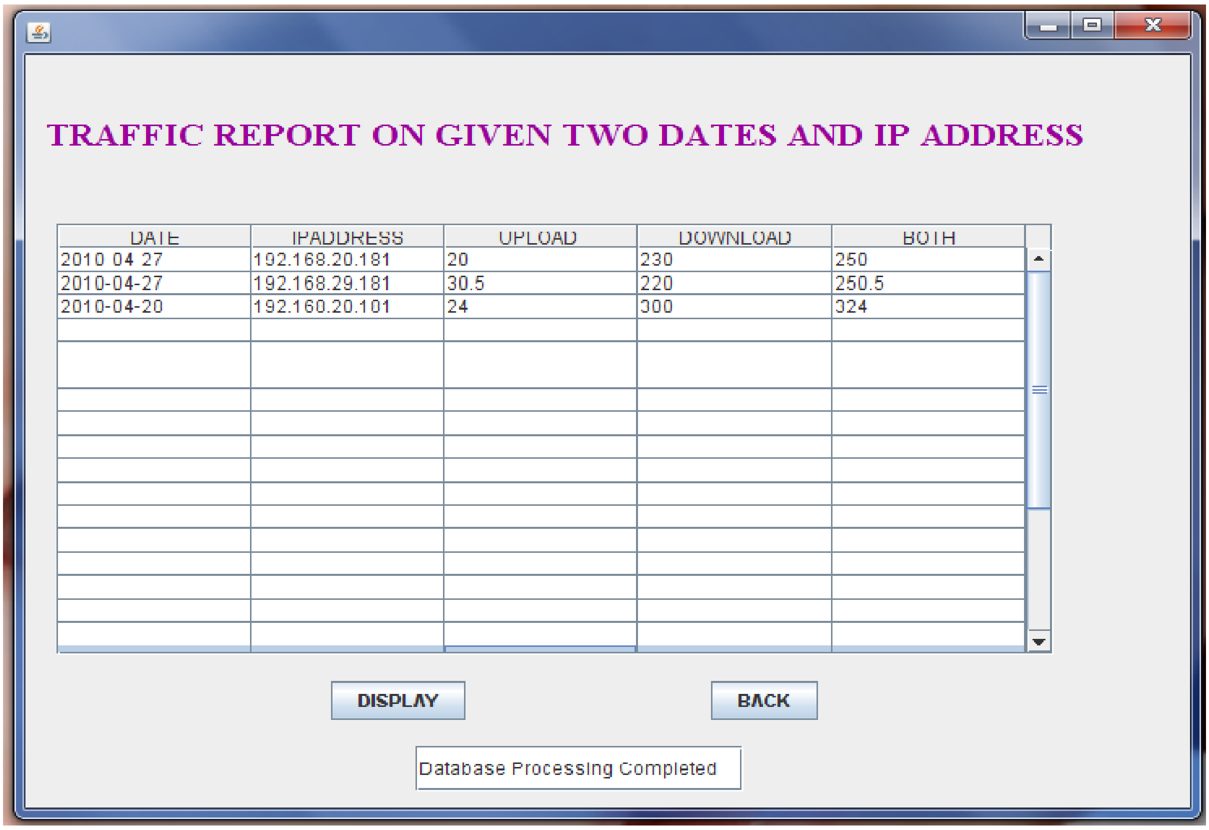
Enter IP Address 192.168.20.102



TRAFFIC REPORT ON GIVEN TWO DATES AND IP ADDRESS

| DATE | IPADDRESS | UPLOAD | DOWNLOAD | BOTH |
|---|---|---|---|---|

DATA PROCESSING......



TRAFFIC REPORT ON GIVEN TWO DATES AND IP ADDRESS

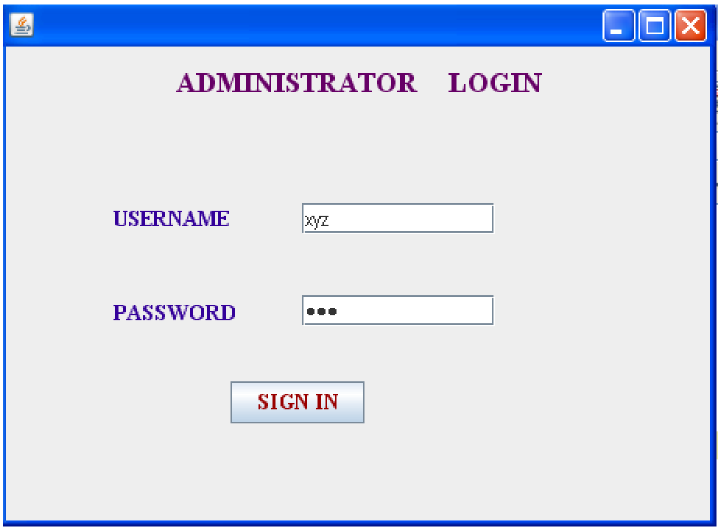
| DATE | IPADDRESS | UPLOAD | DOWNLOAD | BOTH |
|---|---|---|---|---|
| 2010 04 27 | 192.168.20.181 | 20 | 230 | 250 |
| 2010-04-27 | 192.168.29.181 | 30.5 | 220 | 250.5 |
| 2010-04-20 | 192.160.20.101 | 24 | 300 | 324 |

Database Processing Completed

## TEST CASES

A) USERNAME AND PASSWORD CHECKING MODULE.

1) Input: Invalid Username and Password Entered.

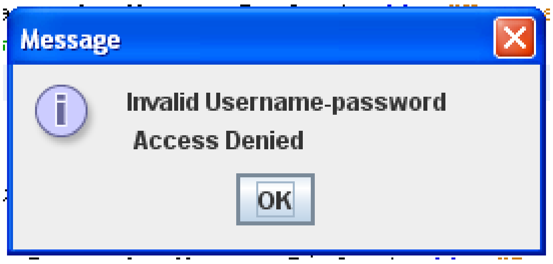


ADMINISTRATOR LOGIN

USERNAME xyz

PASSWORD •••

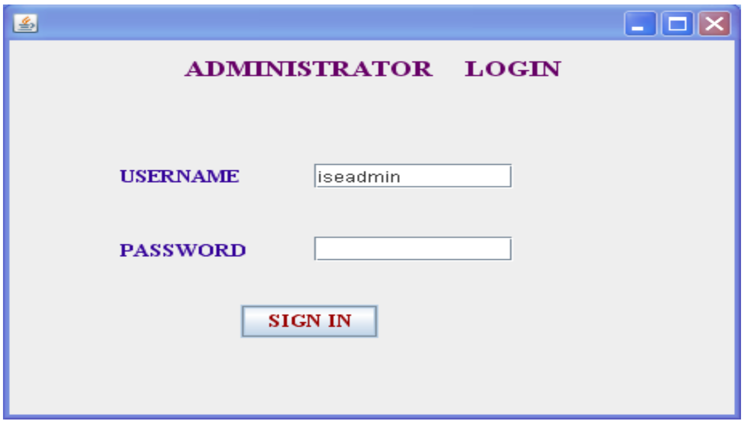
Output: An Alert message is given to user to verify username and password entered.



Message

Invalid Username-password
Access Denied

2) Input: If Password not Entered.

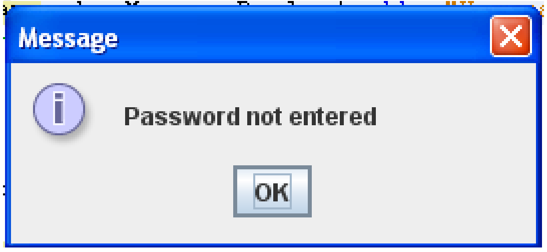


ADMINISTRATOR LOGIN

USERNAME iseadmin

PASSWORD

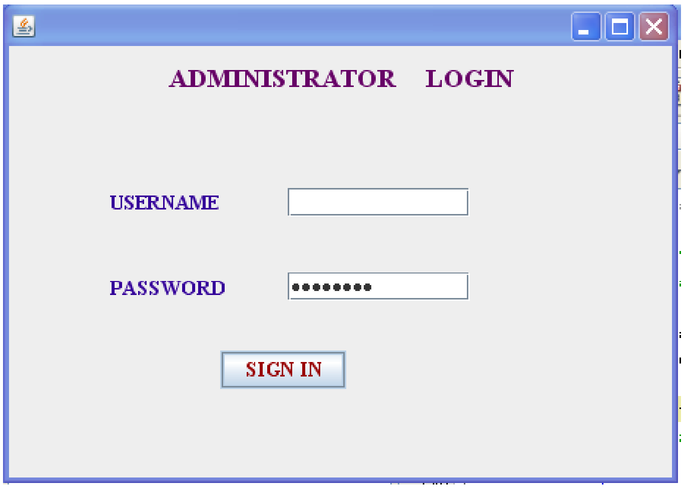
Output: User gets an Alert message to enter the password  before and then proceed.



Message

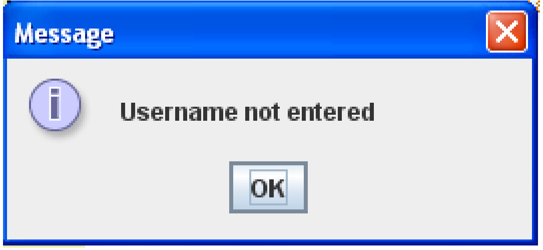
Password not entered

3) Input: If Username not Entered.



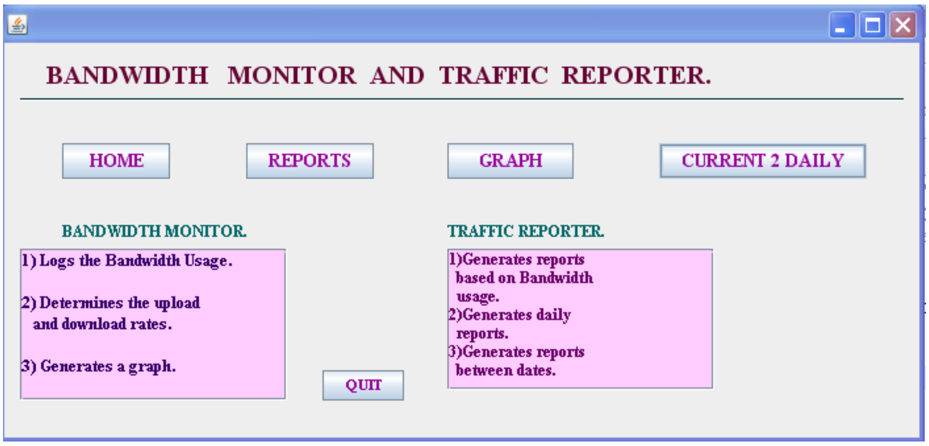
PASSWORD: •••••••••

Output: User gets an Alert message to enter the Username before and then proceed.



Username not entered

4) Input: If Valid username and password is entered by user. Homepage appears.



BANDWIDTH MONITOR.

1) Logs the Bandwidth Usage.

2) Determines the upload and download rates.

3) Generates a graph.

TRAFFIC REPORTER.

1)Generates reports based on Bandwidth usage.

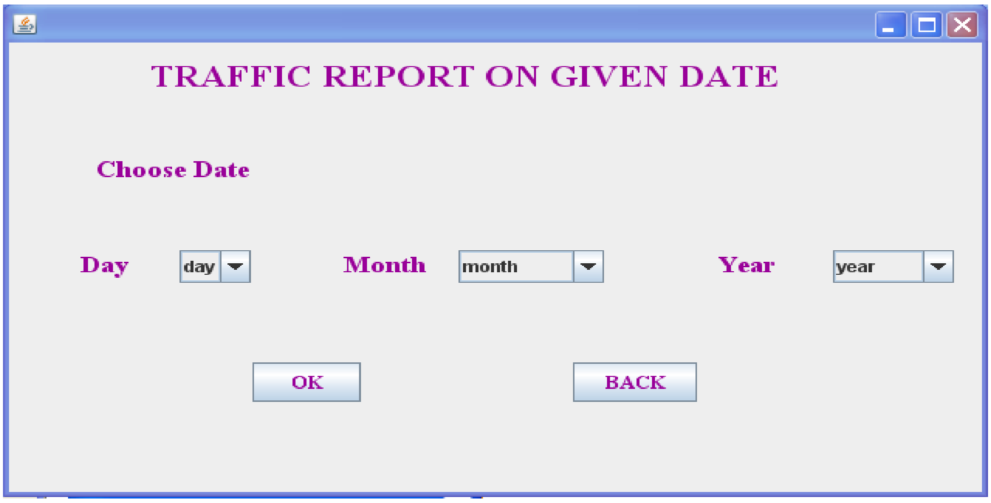
2)Generates daily reports.

3)Generates reports between dates.

B) TRAFFIC REPORTS MODULE.

1) Input: If Day or Month or Year is not selected by user.

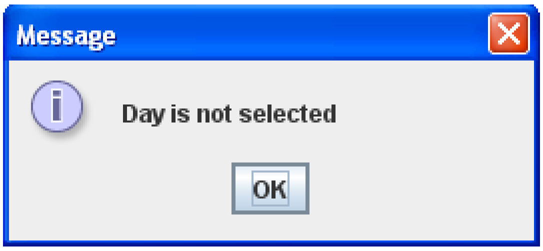
Output: Corresponding Alert messages are given.



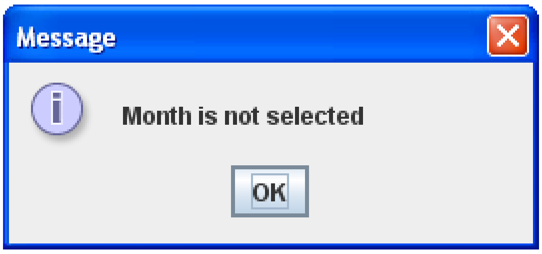
TRAFFIC REPORT ON GIVEN DATE

Choose Date

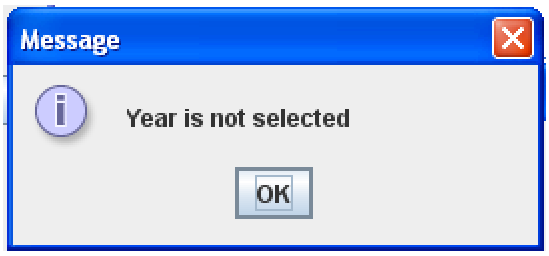
Day day Month month Year year



Day is not selected

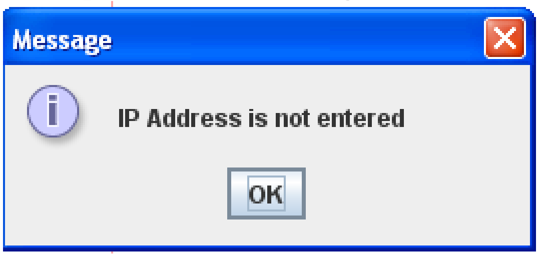


Month is not selected



Year is not selected

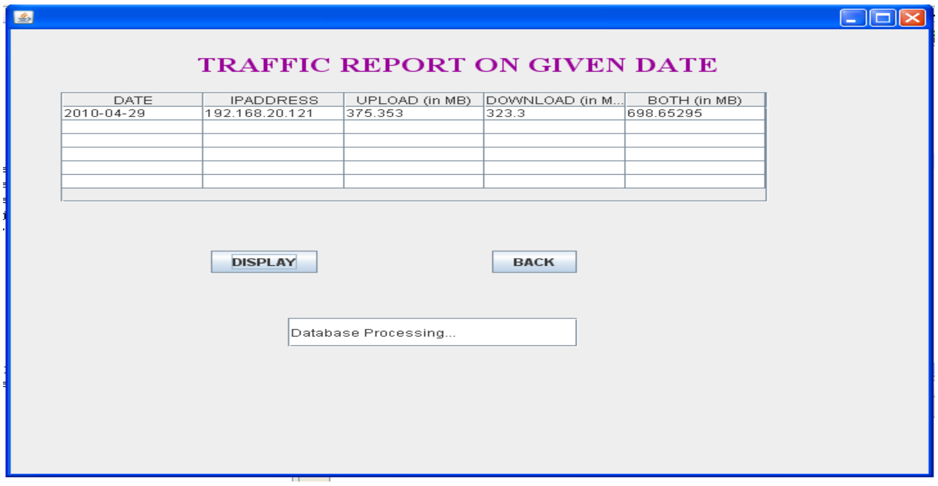
2) Input: If IP Address is not entered by user.
Output: An Alert message of the below form is given to user.



IP Address is not entered

3)Input: On Proper input of day, month and year values (and IP Address).

Output: Report table is Displayed.



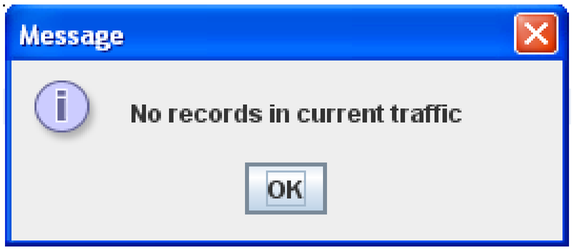
TRAFFIC REPORT ON GIVEN DATE

| DATE | IPADDRESS | UPLOAD (in MB) | DOWNLOAD (in M... | BOTH (in MB) |
|---|---|---|---|---|
| 2010-04-29 | 192.168.20.121 | 375.353 | 323.3 | 698.65295 |

Database Processing...

4) Input: User Clicks on the CurrentToDailyTraffic Button when there are no records in the Current_Traffic table.

Output: An Alert message is given as shown below.

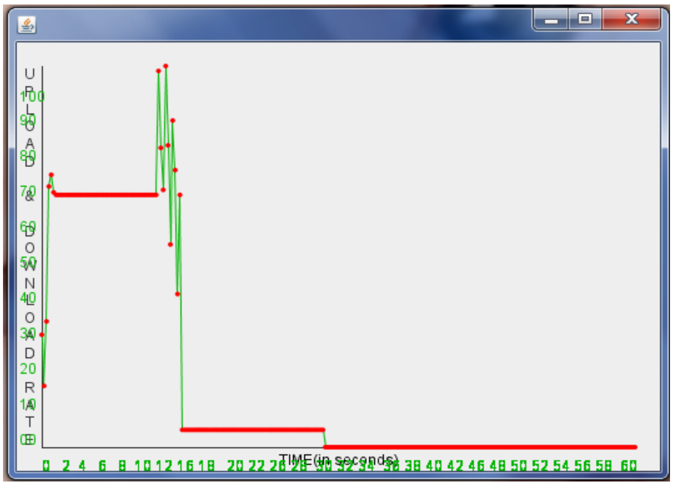


No records in current traffic

C) GRAPH FOR UPLOAD DOWNLOAD RATE AND BOTH.

1) Input: Click on UPLOADRATE or DOWNLOADRATE or BOTH buttons.

Output: Corresponding Graph will be displayed.



# CHAPTER 4

## FUTURE SCOPE:

- This study includes the existing system.
- The study will also help the user to effectively understand the bandwidth usage and the traffic reports that are generated
- The graphical form representation will give the user the clear view of the traffic rates.
- It also covers the disadvantages of the present system.

## REFERENCES:

### Text books:

1. Pankaj Jalote-An Integrated Approach to Software Engineering,3rd edition, Narosa publications-2005

2. Sommerville, I Software Engineering 8th Edition, Pearson Education,2006

3. Java Complete Rereference 4th Edition, Herbeldt Sheild ,2005

### Websites

1. www.snifferheader.com

2. www.bmonitor.com

3. http://netresearch.ics.uci.edu/kfujii/jpcap/doc