**Plugged**

***Form1:***

1. ***Creating a shortcut in the start up folder when the app is executed for the first time:***

if (IsRunningAsAdministrator() && !System.IO.File.Exists(@"C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\plugged.lnk"))

CreateShortcut("plugged", @"C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\", Application.ExecutablePath);

}

catch (Exception exc) { }

1. ***Existence of “test.txt” ensures that the shortcut has been created:***

if (!System.IO.File.Exists(Application.StartupPath+@"\test.txt"))

{

**// Setting up start info of the new process of the same application**

ProcessStartInfo processStartInfo = new ProcessStartInfo(Assembly.GetEntryAssembly().CodeBase);

**// Using operating shell and setting the ProcessStartInfo.Verb to “runas” will let it run as admin**

processStartInfo.UseShellExecute = true;

processStartInfo.Verb = "runas";

**// Start the application as new process**

Process.Start(processStartInfo);

**//Creating the file test.txt to ensure that shortcut will not be created more than once**

StreamWriter w11 = System.IO.File.AppendText(Application.StartupPath + @"\test.txt");

w11.Close();

if (ConnectWIFI != null)

{

ConnectWIFI.Visible = false;

ConnectWIFI.Icon = null; **// required to make icon disappear**

ConnectWIFI.Dispose();

ConnectWIFI = null;

}

//Form.ActiveForm.Close();

Application.Exit();

**// Shut down the current (old) process**

}

1. ***Creating the exit menu for the app :***

ContextMenu c = new ContextMenu(new MenuItem[]{new MenuItem("Exit",Exit)});

ConnectWIFI.ContextMenu = c;

1. ***Checking whether the certificates have been downloaded:***

bool exists1 = System.IO.File.Exists(Application.StartupPath+@"\ca.der");

bool exists2 = System.IO.File.Exists(Application.StartupPath + @"\user2@pl.com.p12");

if (exists1 && exists2)

flag5 = true;

}

1. ***Write the accounting data in a file:***

public void AccWrite()

{

**//Creating a new thread for checking internet connection**

th2 = new Thread(CheckConnection);

th2.Start();

while(true)

{

try

{

th2.Abort();

CurrentWifi cw = new CurrentWifi();

**//flag7 is set to denote the wifi has been disconnected and currentssid != cw.display\_current() denotes that the user has changed the network, in either case ,the data statistics should be written in the file**

if (flag7 || currentssid != cw.display\_current())

{

**//No accounting should be done when the app runs for the first time or the wifi is disconnected. In the former case, currentssid will be empty as checked by the next if condition**

if (!currentssid.Equals("") & !currentssid.Equals("Disconnected"))

System.IO.File.AppendAllText(Application.StartupPath + @"\test.txt", currentssid + " " + bytesent + " " + byterec + Environment.NewLine);

}

currentssid = cw.display\_current();

bytesent = ns.UploadedBytes();

byterec = ns.DownloadedBytes();

}

finally

{ }

flag7 = false;

th2 = new Thread(CheckConnection);

th2.Start();

**//accwrite thread runs after every minute**

Thread.Sleep(60000);

}

}

public Task CallAccWrite()

{

return Task.Run(() =>

{

th1 = new Thread(AccWrite);

th1.Start();

return;

});

}

public async void CallAccWriteTask()

{

await CallAccWrite();

}

1. ***Fetching the data consumed in a particular session:***

public void AccRead()

{

double total = 0.0;

total = (ns.DownloadedBytes() + ns.UploadedBytes()) / 1024.0;

if (total > 1024.0)

{

total = total / 1024.0;

if (total > 1024)

{

total = total / 1024.0;

total = Math.Round(total, 2);

res = total + " GB";

}

else

{

total = Math.Round(total, 2);

res = total + " MB";

}

}

else

{

total = Math.Round(total, 2);

res = total + " KB";

}

}

public Task CallAccRead(Label l1)

{

return Task.Run(() =>

{

th2 = new Thread(AccRead);

th2.Start();

Thread.Sleep(8000);

l1.Text ="Data Used : "+ res;

return;

});

}

public async void CallAccReadTask(Label l1)

{

await CallAccRead(l1);

}

1. ***Creating shortcut:***

public static void CreateShortcut(string shortcutName, string shortcutPath, string targetFileLocation)

{

string shortcutLocation = System.IO.Path.Combine(shortcutPath, shortcutName + ".lnk");

WshShell shell = new WshShell();

IWshShortcut shortcut = (IWshShortcut)shell.CreateShortcut(shortcutLocation);

shortcut.Description = "Plugged"; **// The description of the shortcut**

shortcut.IconLocation = @"c:\myicon.ico"; **// The icon of the shortcut**

shortcut.TargetPath = targetFileLocation; **// The path of the file that will launch when the**

**shortcut is run**

shortcut.Save();

}

1. ***Checking whether the app has been started in administrator mode:***

public static bool IsRunningAsAdministrator()

{

**// Get current Windows user**

WindowsIdentity windowsIdentity = WindowsIdentity.GetCurrent();

**// Get current Windows user principal**

WindowsPrincipal windowsPrincipal = new WindowsPrincipal(windowsIdentity);

**// Return TRUE if user is in role "Administrator"**

return windowsPrincipal.IsInRole(WindowsBuiltInRole.Administrator);

}

1. ***Code for exit option, when the user wants to close the app:***

public void Exit(object sender, EventArgs e)

{

try

{

if (ConnectWIFI != null)

{

ConnectWIFI.Visible = false;

ConnectWIFI.Icon = null; **// required to make icon disappear**

ConnectWIFI.Dispose();

ConnectWIFI = null;

}

}

catch (Exception ex)

{

// handle the error

}

Application.Exit();

}

1. ***Code executed when the form loads:***

private void Form1\_Load(object sender, EventArgs e)

{

//Control.CheckForIllegalCrossThreadCalls = false;

ConnectWIFI.ShowBalloonTip(5);

button1\_Click(sender, e);

}

1. ***Placing the app in the bottom right corner of the windows screen:***

protected override void OnLoad(EventArgs e)

{

Control.CheckForIllegalCrossThreadCalls = false;

this.ShowInTaskbar = false;

PlaceLowerRight();

base.OnLoad(e);

}

private void PlaceLowerRight()

{

**//Determine "rightmost" screen**

Screen rightmost = Screen.AllScreens[0];

foreach (Screen screen in Screen.AllScreens)

{

if (screen.WorkingArea.Right > rightmost.WorkingArea.Right)

rightmost = screen;

}

this.Left = rightmost.WorkingArea.Right - this.Width;

this.Top = rightmost.WorkingArea.Bottom - this.Height;

}

protected override void WndProc(ref Message m)

{

const int WM\_SYSCOMMAND = 0x0112;

const int SC\_MOVE = 0xF010;

switch (m.Msg)

{

case WM\_SYSCOMMAND:

int command = m.WParam.ToInt32() & 0xfff0;

if (command == SC\_MOVE)

return;

break;

}

base.WndProc(ref m)

}

1. ***Checking whether the connection has been dropped (disconnected):***

public void CheckConnection()

{

while (true)

{

CurrentWifi cw = new CurrentWifi();

String status = cw.display\_current();

if (status.Equals("Disconnected"))

{

flag7 = true;

if (th3 != null)

th3.Abort();

}

}

}

1. ***Executed when the form loads:***

private async void button1\_Click(object sender, EventArgs e)

{

panel1.Visible = false;

panel2.Visible = true;

String id = wifistatus();

if (id.Equals("Disabled")) **//if wifi adapter is disabled**

{

//wifilist.Visible = false;

labelX1.Visible = true;

enable.Visible = true;

}

else **//else show the list of available wifis**

{

circularProgress1.IsRunning = true;

circularProgress1.Visible = true;

await Task.Delay(3000);

this.Invoke(new List\_of\_wifi\_thread(this.List\_of\_wifi), sender, e);

// CallAccWriteTask();

}

}

1. ***Displaying the list of wifis:***

private async void List\_of\_wifi(object sender, EventArgs e)

{

Label l1 = new Label();

Label l2 = new Label();

Label l = new Label();

PictureBox picbox;

Button dis = new Button();

flag4 = false;

panel3.Controls.Clear();

String id = wifistatus();

int xPos = 4; **//coordinates of wifi buttons(showing individual ssids)**

int yPos = 4;

WifiInfo wifi = new WifiInfo();

string[] profiles = wifi.ssidList(); **//ssid list**

int[] signal = wifi.signalStrengths(); **//list of signal strengths**

string[] mode = wifi.authTypes(); **//list of authentication mode**

string[] bssid = wifi.bssidList(); **//bssid list**

length = profiles.Length; **//length of the ssid list**

***//sorting the wifis based on signal strength***

int temp = 0;

string temp1 = "";

for (int write = 0; write < length; write++)

{

for (int sort = 0; sort < length - 1; sort++)

{

if (signal[sort] < signal[sort + 1])

{

temp = signal[sort + 1];

signal[sort + 1] = signal[sort];

signal[sort] = temp;

temp1 = profiles[sort + 1];

profiles[sort + 1] = profiles[sort];

profiles[sort] = temp1;

temp1 = mode[sort + 1];

mode[sort + 1] = mode[sort];

mode[sort] = temp1;

temp1 = bssid[sort + 1];

bssid[sort + 1] = bssid[sort];

bssid[sort] = temp1;

}

}

}

**//array of available wifis**

Button[] btnArray = new Button[length];

int dist = 0; **//variable for incrementing the coordinates of tools**

int xpic = 333;

int ypic = 19;

int xlabel = 237; **//coordinates of “connected” label**

int ylabel = 13;

int xlabel1 = 9; **//coordinates of “ Users connected” label**

int ylabel1 = 50;

int xlabel2 = 9; **//coordinates of “Data used” label**

int ylabel2 = 35;

int xdisc = 242; **//coordinates of disconnect button**

int ydisc = 43;

**//populating the list**

for (int i = 0; i < length; i++)

{

int h1 = 5; **//height and weight of wifi signals**

int w1 = 3;

int x = 320; **//coordinates of wifi signal**

int y = 23 + dist;

btnArray[i] = new System.Windows.Forms.Button();

**//creating wifi signals**

Button[] r = new Button[5];

for (int j = 0; j < 5; j++)

{

r[j] = new Button();

r[j].Height = h1;

r[j].Width = w1;

r[j].Top = y;

r[j].Left = x;

r[j].FlatStyle = FlatStyle.Flat;

r[j].BackColor = Color.Silver;

r[j].FlatAppearance.BorderSize = 0;

btnArray[i].FlatAppearance.BorderColor = Color.FromArgb(0, 255, 255, 255);

r[j].Visible = true;

h1 += 4;

x += 4;

y -= 4;

wifilist.Controls.Add(r[j]);

}

**//filling the signals with colours according to their signal strength**

if (signal[i] >= 0 && signal[i] <= 20)

{

r[0].BackColor = Color.MediumSeaGreen;

}

else if (signal[i] > 20 && signal[i] <= 40)

{

r[0].BackColor = Color.MediumSeaGreen;

r[1].BackColor = Color.MediumSeaGreen;

}

else if (signal[i] > 40 && signal[i] <= 60)

{

r[0].BackColor = Color.MediumSeaGreen;

r[1].BackColor = Color.MediumSeaGreen;

r[2].BackColor = Color.MediumSeaGreen;

}

else if (signal[i] > 60 && signal[i] <= 80)

{

r[0].BackColor = Color.MediumSeaGreen;

r[1].BackColor = Color.MediumSeaGreen;

r[2].BackColor = Color.MediumSeaGreen;

r[3].BackColor = Color.MediumSeaGreen;

}

else

{

r[0].BackColor = Color.MediumSeaGreen;

r[1].BackColor = Color.MediumSeaGreen;

r[2].BackColor = Color.MediumSeaGreen;

r[3].BackColor = Color.MediumSeaGreen;

r[4].BackColor = Color.MediumSeaGreen;

}

btnArray[i].Width = 350; **// size of wifi button**

btnArray[i].Left = xPos;

btnArray[i].Top = yPos;

btnArray[i].Text = profiles[i]; **//profiles[i] contains the ssid of the network**

btnArray[i].ForeColor = Color.RoyalBlue;

btnArray[i].TabStop = false;

**//if profiles[i] contains the ssid of the connected network and is an enterprise one**

if (id.Equals(profiles[i]) && mode[i].Equals("WPA2-Enterprise"))

{

btnArray[i].Font = new Font("Cambria", 12, FontStyle.Bold);

btnArray[i].Height = 72;

btnArray[i].TextAlign = ContentAlignment.TopLeft;

yPos += 72;

dist += 72; **//used for wifi signal**

ToolTip tt1=new ToolTip();

**//Mouse hover event-a parameterized call**

btnArray[i].MouseEnter += (sender1, e1) => { EntButtonMouseEnter(sender1, e1, l, l1, l2, tt1); };

btnArray[i].MouseLeave += (sender1, e1) => { EntButtonMouseLeave(sender1, e1, l, l1, l2, tt1); };

}

**// if profiles[i] contains the ssid of the connected network and is not an enterprise one**

else if (id.Equals(profiles[i]))

{

btnArray[i].Font = new Font("Cambria", 12, FontStyle.Bold);

btnArray[i].Height = 72;

btnArray[i].TextAlign = ContentAlignment.TopLeft;

yPos += 72;

dist += 72; **//used for wifi signal**

btnArray[i].MouseEnter += (sender1, e1) => {ButtonMouseEnter(sender1, e1, l, l1, l2, dis); };

btnArray[i].MouseLeave += (sender1, e1) => { ButtonMouseLeave(sender1, e1, l, l1, l2, dis); };

}

**//if profiles[i] is not the same as connected network**

else

{

btnArray[i].Font = new Font("Cambria", 12);

//btnArray[i].ForeColor = Color.MediumSeaGreen;

btnArray[i].Height = 30;

btnArray[i].TextAlign = ContentAlignment.MiddleLeft;

yPos += 30;

dist += 30; **//used for wifi signal**

}

btnArray[i].BackColor = Color.White;

btnArray[i].FlatStyle = FlatStyle.Flat;

btnArray[i].FlatAppearance.BorderSize = 0;

btnArray[i].FlatAppearance.BorderColor = Color.FromArgb(0, 255, 255, 255);

**//Defining the click event of buttons**

**//if profiles[i] and the current network is not same**

if (!id.Equals(profiles[i]))

{

if (mode[i].Equals("WPA2-Enterprise")) **//and is an enterprise one**

{

btnArray[i].Click += new System.EventHandler(WifiConnect\_enterprise);

ToolTip tt1 = new ToolTip();

btnArray[i].MouseEnter += (sender1, e1) => { Stats\_show(sender1, e1, tt1); };

btnArray[i].MouseLeave += (sender1, e1) => { Stats\_show(sender1, e1, tt1); };

}

else **//and not an enterprise one**

{

btnArray[i].Click += new System.EventHandler(WifiConnect);

}

}

**//else profiles[i] and the current network is same**

else

{

**//imposing a firewall rule**

if (flag6)

{

url = "www.stackoverflow.com";

Firewall fw = new Firewall();

//fw.SiteBlock(url);

}

**//”Connected” label**

l.Width = 75;

l.Height = 15;

l.Top = ylabel;

l.Left = xlabel;

l.BackColor = Color.Transparent;

l.ForeColor = Color.DarkBlue;

l.Text = "Connected";

l.Font = new Font("Cambria", 10,FontStyle.Bold|FontStyle.Italic);

panel3.Controls.Add(l);

**//”Users connected “label**

l1.Width = 185;

l1.Height = 15;

l1.Top = ylabel1;

l1.Left = xlabel1;

l1.BackColor = Color.Transparent;

l1.ForeColor = Color.DarkBlue;

l1.Text = "Users Connected : Calculating.....";

l1.Font = new Font("Cambria", 9, FontStyle.Italic);

panel3.Controls.Add(l1);

noofusers(l1); **//For calculating the no of users by calling a thread**

**//”Data used” label**

l2.Width = 185;

l2.Height = 15;

l2.Top = ylabel2;

l2.Left = xlabel2;

l2.BackColor = Color.Transparent;

l2.ForeColor = Color.DarkBlue;

l2.Text = "Data Used : Calculating.......";

l2.Font = new Font("Cambria", 9, FontStyle.Italic);

ToolTip tt1 = new ToolTip();

l2.MouseEnter += (sender1, e1) => { DataMouseEnter(sender1, e1,l2, tt1); };

l2.MouseLeave += (sender1, e1) => { DataMouseLeave(sender1, e1, l2, tt1); };

panel3.Controls.Add(l2);

CallAccReadTask(l2); **//For calculating the data consumed by calling a thread**

**//Disconnect button**

dis.Height = 26;

dis.Width = 98;

dis.Top = ydisc;

dis.Left = xdisc;

dis.FlatStyle = FlatStyle.Standard;

dis.FlatAppearance.BorderSize = 1;

dis.Text = "Disconnect";

dis.Font = new Font("Cambria", 10, FontStyle.Bold);

dis.Click += new System.EventHandler(WifiDisconnect);

panel3.Controls.Add(dis);

}

flag6 = false;

**//incrementing the coordinates of buttons and labels**

ydisc += 30;

ylabel += 30;

ylabel1 += 30;

ylabel2 += 30;

Label l11=new Label();

if (bssid[i].Equals("Multiple")) **//if multiple access points of the network are available**

{

**//creating the icon**

picbox= new PictureBox();

picbox.BackColor = Color.Transparent;

picbox.Image = Properties.Resources.sharegreen;

picbox.Location = new Point(xpic, ypic);

picbox.Size = new Size(10, 10);

picbox.SizeMode = PictureBoxSizeMode.StretchImage;

picbox.Visible = true;

picbox.BringToFront();

panel3.Controls.Add(picbox);

**//if it is the current network, then record the ssid and bssid of the network for auto-switching**

if (id.Equals(profiles[i]))

{

Multiple\_ssid = profiles[i];

Multiple\_bssid = WifiInfo.multiplebssid;

//MultipleAccessPoint();

}

}

**//if the current network is same as profiles[i],then the coordinate’s value is increased by 72 because the**

**width of button will be more, else the value is increased by 30**

if (id.Equals(profiles[i]))

ypic+=72;

else

ypic += 30;

**//wifi button is added to the panel**

panel3.Controls.Add(btnArray[i]);

**//wifi signals for that wifi are added to the panel**

for (int k = 0; k < 5; k++)

{

panel3.Controls.Add(r[k]);

}

btnArray[i].SendToBack();

}

**//if wifi is connected**

if (!id.Equals("Disconnected"))

{

wifiname.Text = "Currently Connected To : "; **//show the name of the network**

ssid.Text = id;

**//show internet problems if any ,else show “Internet Access”**

internetconnection.Text = "( " + st.CheckInternetConnection() + " )";

**//if internet is working properly then check the quality of connection**

if (internetconnection.Text.Equals("( Internet Access )") == true)

{

String quality = st.PingTimeAverage("www.stackoverflow.com", 4);

linequality.Text = quality;

}

else

linequality.Text = "Poor Speed";

ssid.BringToFront();

internetconnection.BringToFront();

linequality.BringToFront();

}

else **//wifi is not connected**

{

wifiname.Text = "Not Connected";

labelX4.BringToFront();

}

await Task.Delay(3000);

**//the wifi list is ready to be displayed**

circularProgress1.IsRunning = false;

circularProgress1.Visible = false;

circularProgress2.IsRunning = false;

circularProgress2.Visible = false;

panel2.Visible = true;

**//if wifi is enabled then show the appropriate labels and images**

if (!id.Equals("Disconnected"))

{

labelX4.Visible = false;

ssid.Visible = true;

internetconnection.Visible = true;

linequality.Visible = true;

**//if internet is working fine ,then only the user can check the speed**

if (internetconnection.Text.Equals("( Internet Access )")==true)

CheckSpeed.Enabled = true;

else

CheckSpeed.Enabled = false;

CheckSpeed.Visible = true;

netunavailable.Visible = false;

netavailable.BringToFront();

netavailable.Visible = true;

}

else

{

ssid.Visible = false;

internetconnection.Visible = false;

linequality.Visible = false;

CheckSpeed.Visible = false;

labelX4.Visible = true;

netavailable.Visible = false;

netunavailable.BringToFront();

netunavailable.Visible = true;

}

wifiname.Visible = true;

refresh.Visible = true;

wifilist.Visible = true;

panel3.Visible = true;

}

1. ***To check the no of users connected by creating an async task:***

public async void noofusers(Label l1)

{

await account(l1);

}

public Task account(Label l1)

{

return Task.Run(() =>

{

Thread worker;

String host1 = st.GetHost1(); **//get the class of the ip address,eg. 192.168.1.**

for (j1 = 1; j1 <= 255; j1++) **//creating individual threads to ping every device on the network**

{

host =host1+j1;

worker = new Thread(StartAcc);

worker.Start();

}

Thread.Sleep(4000);

users = st.GetHost2(); **//getting the count**

l1.Text = "Users Connected : " + users;

return;

});

}

1. ***Method to ping the host:***

public void StartAcc()

{

acc.PingHost(host);

}

1. ***To check if another access point is giving a better signal than the one connected(in case of multiple access points):***

public async void MultipleAccessPoint()

{

await MultipleAccessPointThread();

}

public Task MultipleAccessPointThread()

{

return Task.Run(() =>

{

return Task.Run(() =>

{

if (th3 != null)

th3.Abort();

th3 = new Thread(MultipleAccessPointSignal);

th3.Start();

return;

});

});

}

public async void MultipleAccessPointSignal()

{

bool signal=false;

while (true)

{

signal = acc.MultipleAccessPoint(Multiple\_ssid, Multiple\_bssid);

**//if a better access point is available then notify the user**

if (!signal)

{

Form4 f4 = new Form4();

f4.ssid1 = Multiple\_ssid;

ConnectWIFI.Visible = false;

f4.timerstart();

Hide();

f4.ShowDialog();

await Task.Delay(1000);

ConnectWIFI.Visible = true;

Show();

this.Activate();

}

Thread.Sleep(300000);

}

}

1. ***Mouse hover text for “data used” label:***

private void DataMouseEnter(Object sender, EventArgs e, Label l2, ToolTip tt1)

{

tt1.InitialDelay = 0;

Label p = (Label)sender;

//p.BackColor = Color.Lavender;

tt1.Show("Data consumed in this session", p, 0);

//flag4 = true;

}

private void DataMouseLeave(Object sender, EventArgs e, Label l2, ToolTip tt1)

{

Label p = (Label)sender;

if (!this.ClientRectangle.Contains(this.PointToClient(Cursor.Position)))

{

tt1.Hide(p);

}

}

1. ***Mouse hover event for enterprise mode(currently connected):***

private void EntButtonMouseEnter(Object sender, EventArgs e, Label l, Label l1, Label l2, ToolTip tt1)

{

tt1.InitialDelay = 0;

Button p = (Button)sender;

//p.BackColor = Color.Lavender;

if (!flag4)

{

tt1.Show(msg, p, 0);

flag4 = true;

}

l.BackColor = Color.Lavender;

l1.BackColor = Color.Lavender;

l2.BackColor = Color.Lavender;

}

private void EntButtonMouseLeave(Object sender, EventArgs e, Label l, Label l1, Label l2, ToolTip tt1)

{

Button p = (Button)sender;

if (!this.ClientRectangle.Contains(this.PointToClient(Cursor.Position)))

{

tt1.Hide(p);

flag4 = false;

}

//p.BackColor = Color.Transparent;

l.BackColor = Color.Transparent;

l1.BackColor = Color.Transparent;

l2.BackColor = Color.Transparent;

}

1. ***Mouse hover event for normal mode(currently connected):***

private void ButtonMouseEnter(Object sender, EventArgs e, Label l, Label l1, Label l2, Button dis)

{

Button p = (Button)sender;

//p.BackColor = Color.Lavender;

l.BackColor = Color.Lavender;

l1.BackColor = Color.Lavender;

l2.BackColor = Color.Lavender;

}

private void ButtonMouseLeave(Object sender, EventArgs e, Label l, Label l1, Label l2, Button dis)

{

Button p = (Button)sender;

//p.BackColor = Color.Transparent;

l.BackColor = Color.Transparent;

l1.BackColor = Color.Transparent;

l2.BackColor = Color.Transparent;

}

1. ***Tooltip to show “shared wifi” in case of enterprise mode:***

private void Stats\_show(Object sender, EventArgs e,ToolTip tt1)

{

tt1.InitialDelay = 0;

Button p = (Button)sender;

if (!flag4)

{

tt1.Show(msg, p, 0);

flag4 = true;

}

}

private void Stats\_hide(Object sender, System.EventArgs e, ToolTip tt1)

{

Button p = (Button)sender;

// p.BackColor = Color.Transparent;

if (!this.ClientRectangle.Contains(this.PointToClient(Cursor.Position)))

{

tt1.Hide(p);

flag4 = false;

}

}

1. ***To connect to an enterprise network:***

public async void WifiConnect\_enterprise(Object sender, System.EventArgs e)

{

if (!flag5) **//if the certificates are not present in the specified folder, we have to download it**

{

Form5 f2 = new Form5();

Form3 f = new Form3();

f2.LabelText = "Enter The Company Security Key";

f2.setmode = true;

Hide();

ConnectWIFI.Visible = false;

await Task.Delay(2000);

f2.ShowDialog();

ConnectWIFI.Visible = true;

await Task.Delay(1000);

if (f2.Okbuttonclicked == true)

{

security\_key = f2.secretkey; **//verification part still needs to be done**

System.Net.WebClient wc = new System.Net.WebClient();

**//download+install**

wc.DownloadFile("http://52.27.54.85/blog/rad\_app/uploads/cert.php?filename=ca.der", Application.StartupPath + @"\ca.der");

wc.DownloadFile("http://52.27.54.85/blog/rad\_app/uploads/cert.php?filename=user2@pl.com.p12", Application.StartupPath + @"\user2@pl.com.p12");

//Cert cert = new Cert();

cert.Add\_der(Application.StartupPath + @"\ca.der");

cert.Add\_p12(Application.StartupPath + @"\user2@pl.com.p12", "clientkey");

Button btn = (Button)sender;

**//xml profile is being created**

xml.GnerateWPA2tls(btn.Text, Application.StartupPath + @"\" + btn.Text + ".xml");

**//the profile is added to the profile list**

netsh.ProfileAdd(Application.StartupPath+@"\" + btn.Text + ".xml");

flag5 = true;

f.LabelText = "Connecting to " + btn.Text + " .........";

ConnectWIFI.Visible = false;

Hide();

//await Task.Delay(400);

f.Show();

await Task.Delay(2000);

String res = netsh.Connect(btn.Text);

**//if wifi connection has been done successfully**

if (!res.Contains("successfully"))

{

f.Hide();

Form3 f1 = new Form3();

f1.LabelText = "Windows was unable to connect to " + btn.Text;

ConnectWIFI.Visible = false;

f1.ShowDialog();

ConnectWIFI.Visible = true;

}

else

{

//wifiname.Visible = false;

**//Again ensure that the wifi connection has been done successfully**

CurrentWifi cw = new CurrentWifi();

if (cw.display\_current().Equals(btn.Text))

{

//if (url.Equals(""))

//{

// url = "www.facebook.com";

// Firewall fw = new Firewall();

// fw.SiteBlock(url);

//}

//else

//{

// Firewall fw = new Firewall();

// fw.SiteUnblock(url);

// url = "";

//}

panel3.Visible = false;

//ssid.Visible = false;

//refresh.Visible = false;

//wifilist.Visible = false;

circularProgress2.Visible = true;

circularProgress2.IsRunning = true;

//repopulate the wifi list

List\_of\_wifi(sender, e);

}

else

{

f.Hide();

Form3 f1 = new Form3();

f1.LabelText = "Windows was unable to connect to " + btn.Text;

ConnectWIFI.Visible = false;

f1.ShowDialog();

ConnectWIFI.Visible = true;

}

}

await Task.Delay(5000);

ConnectWIFI.Visible = true;

Show();

this.Activate();

}

else

{

ConnectWIFI.Visible = true;

Show();

this.Activate();

}

//}

}

else **//certificates are present in the specified folder**

{

Form3 f1 = new Form3();

Button btn = (Button)sender;

**//add the certificates in the “certicates” folder of machine**

cert.Add\_der(Application.StartupPath + @"\ca.der");

cert.Add\_p12(Application.StartupPath + @"\user2@pl.com.p12", "clientkey");

**//check if profile is there in the list, if not ,then generate and add the profile**

if (netsh.CheckProfile(btn.Text) == "not present")

{

xml.GnerateWPA2tls(btn.Text, Application.StartupPath + @"\" + btn.Text + ".xml");

netsh.ProfileAdd(Application.StartupPath + @"\" + btn.Text + ".xml");

}

f1.LabelText = "Connecting to " + btn.Text + " .........";

ConnectWIFI.Visible = false;

Hide();

//await Task.Delay(400);

f1.Show();

await Task.Delay(2000);

//char[] delimeter = { ' ' };

//String[] network\_info = btn.Text.Split(delimeter);

String res = netsh.Connect(btn.Text);

if (!res.Contains("successfully"))

{

f1.Hide();

Form3 f2 = new Form3();

f2.LabelText = "Windows was unable to connect to " + btn.Text;

ConnectWIFI.Visible = false;

f2.ShowDialog();

ConnectWIFI.Visible = true;

}

else

{

//wifiname.Visible = false;

CurrentWifi cw = new CurrentWifi();

if (cw.display\_current().Equals(btn.Text))

{

//if (url.Equals(""))

//{

// url = "www.facebook.com";

// Firewall fw = new Firewall();

// fw.SiteBlock(url);

//}

//else

//{

// Firewall fw = new Firewall();

// fw.SiteUnblock(url);

// url = "";

//}

//panel3.Visible = false;

for (int i = 0; i < 5; i++)

r1[i].Visible = false;

//ssid.Visible = false;

//refresh.Visible = false;

//wifilist.Visible = false;

circularProgress2.Visible = true;

circularProgress2.IsRunning = true;

List\_of\_wifi(sender, e);

}

else

{

f1.Hide();

Form3 f2 = new Form3();

f2.LabelText = "Windows was unable to connect to " + btn.Text;

ConnectWIFI.Visible = false;

f2.ShowDialog();

ConnectWIFI.Visible = true;

}

}

await Task.Delay(5000);

ConnectWIFI.Visible = true;

Show();

this.Activate();

}

}

1. ***To connect to a non-enterprise network:***

public async void WifiConnect(Object sender, System.EventArgs e)

{

Button btn = (Button)sender;

if (netsh.CheckProfile(btn.Text) == "not present")

{

Form5 f2 = new Form5();

Hide();

await Task.Delay(2000);

ConnectWIFI.Visible = false;

f2.ShowDialog();

await Task.Delay(1000);

ConnectWIFI.Visible = true;

if (f2.Okbuttonclicked == true)

{

security\_key = f2.secretkey;

xml.GenerateWPA2psk(btn.Text, security\_key, Application.StartupPath + @"\" + btn.Text + ".xml");

netsh.ProfileAdd(Application.StartupPath + @"\" + btn.Text + ".xml");

Form3 f = new Form3();

f.LabelText = "Connecting to " + btn.Text + " .........";

ConnectWIFI.Visible = false;

Hide();

//await Task.Delay(400);

f.Show();

await Task.Delay(2000);

//char[] delimeter = { ' ' };

//String[] network\_info = btn.Text.Split(delimeter);

String res = netsh.Connect(btn.Text);

if (!res.Contains("successfully"))

{

f.Hide();

Form3 f1 = new Form3();

f1.LabelText = "Windows was unable to connect to " + btn.Text;

ConnectWIFI.Visible = false;

f1.ShowDialog();

ConnectWIFI.Visible = true;

}

else

{

//wifiname.Visible = false;

//if (url.Equals(""))

//{

// url = "www.facebook.com";

// Firewall fw = new Firewall();

// fw.SiteBlock(url);

//}

//else

//{

// Firewall fw = new Firewall();

// fw.SiteUnblock(url);

// url = "";

//}

panel3.Visible = false;

//ssid.Visible = false;

//refresh.Visible = false;

//wifilist.Visible = false;

circularProgress2.Visible = true;

circularProgress2.IsRunning = true;

List\_of\_wifi(sender, e);

}

await Task.Delay(5000);

ConnectWIFI.Visible = true;

Show();

this.Activate();

}

else

{

ConnectWIFI.Visible = true;

Show();

this.Activate();

}

}

else

{

Form3 f = new Form3();

f.LabelText = "Connecting to " + btn.Text + " .........";

ConnectWIFI.Visible = false;

Hide();

//await Task.Delay(400);

f.Show();

await Task.Delay(2000);

//char[] delimeter = { ' ' };

//String[] network\_info = btn.Text.Split(delimeter);

String res = netsh.Connect(btn.Text);

if (!res.Contains("successfully"))

{

f.Hide();

Form3 f1 = new Form3();

f1.LabelText = "Windows was unable to connect to " + btn.Text;

ConnectWIFI.Visible = false;

f1.ShowDialog();

ConnectWIFI.Visible = true;

}

else

{

//wifiname.Visible = false;

//if (url.Equals(""))

//{

// url = "www.facebook.com";

// Firewall fw = new Firewall();

// fw.SiteBlock(url);

//}

//else

//{

// Firewall fw = new Firewall();

// fw.SiteUnblock(url);

// url = "";

//}

panel3.Visible = false;

//ssid.Visible = false;

//refresh.Visible = false;

//wifilist.Visible = false;

circularProgress2.Visible = true;

circularProgress2.IsRunning = true;

await Task.Delay(3000);

List\_of\_wifi(sender, e);

}

await Task.Delay(5000);

ConnectWIFI.Visible = true;

Show();

this.Activate();

}

}

1. ***To disconnect the wifi :***

public async void WifiDisconnect(Object sender, System.EventArgs e)

{

Hide();

//if (!url.Equals(""))

//{

// Firewall fw = new Firewall();

// fw.SiteUnblock(url);

// url = "";

//}

Form3 f = new Form3();

f.LabelText = "Disconnecting from Wifi";

ConnectWIFI.Visible = false;

//await Task.Delay(2000);

//await CallDisconnect(f);

f.Show();

await Task.Delay(2000);

**//after disconnecting show the main form again**

this.Invoke(new List\_of\_wifi\_thread(this.func1), sender, e);

await Task.Delay(5000);

ConnectWIFI.Visible = true;

Show();

this.Activate();

}

public void func1(Object sender, System.EventArgs e)

{

netsh.Disconnect();

panel3.Visible = false;

List\_of\_wifi(sender, e);

}

1. ***To check the ssid of current wifi:***

public string wifistatus()

{

CurrentWifi cw = new CurrentWifi();

return(cw.display\_current());

}

1. ***To show the list of wifi when the wifi adapter is enabled:***

private void enable\_Click(object sender, EventArgs e)

{

labelX1.Visible = false;

enable.Visible = false;

circularProgress1.Visible = true;

circularProgress1.IsRunning = true;

List\_of\_wifi(sender, e);

}

1. ***When “refresh” button is clicked:***

private async void refresh\_Click(object sender, EventArgs e)

{

//panel2.Visible = false;

//msg = null;

panel3.Visible = false;

circularProgress2.IsRunning = true;

circularProgress2.Visible = true;

await Task.Delay(3000);

List\_of\_wifi(sender, e);

}

1. ***When “notify icon” is clicked:***

private void ConnectWIFI\_Click(object sender, EventArgs e)

{

if (flag3 == false) **//if the form is hidden, then clicking on notify icon should display the form**

{

Show();

//refresh\_Click(sender, e);

flag3 = true;

}

else **//else hide(minimize) the form**

{

Hide();

flag3 = false;

}

}

1. ***When the “Check Speed” button is clicked:***

private async void CheckSpeed\_Click(object sender, EventArgs e)

{

Form2 f = new Form2();

Hide();

ConnectWIFI.Visible = false;

await Task.Delay(1000);

f.ShowDialog();

ConnectWIFI.Visible = true;

Show();

this.Activate();

}

1. ***When the user clicks outside the form ,then the form should be minimized:***

private void Form1\_Deactivate(object sender, EventArgs e)

{

Hide();

flag3 = false;

}

1. ***Mouse hover event for “refresh” button:***

private void refresh\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Refresh", (Button)sender);

}

}

}

***Form2:***

1. ***Placing the form in bottom right position:***

protected override void OnLoad(EventArgs e)

{

PlaceLowerRight();

base.OnLoad(e);

}

private void PlaceLowerRight()

{

**//Determine "rightmost" screen**

Screen rightmost = Screen.AllScreens[0];

foreach (Screen screen in Screen.AllScreens)

{

if (screen.WorkingArea.Right > rightmost.WorkingArea.Right)

rightmost = screen;

}

this.Left = rightmost.WorkingArea.Right - this.Width;

this.Top = rightmost.WorkingArea.Bottom - this.Height;

}

protected override void WndProc(ref Message m)

{

const int WM\_SYSCOMMAND = 0x0112;

const int SC\_MOVE = 0xF010;

switch (m.Msg)

{

case WM\_SYSCOMMAND:

int command = m.WParam.ToInt32() & 0xfff0;

if (command == SC\_MOVE)

return;

break;

}

base.WndProc(ref m);

}

1. ***Calculating the speed as soon as the form loads:***

private async void Form2\_Load(object sender, EventArgs e)

{

//await Task.Delay(500);

operation.Text = "Calculating download speed.Please wait.";

circularProgress1.IsRunning = true;

circularProgress1.Visible = true;

await Task.Delay(1000);

String ds = await CalculateSpeed('d'); //calling async task

//await Task.Delay(1000);

circularProgress1.Visible = false;

dspeed.Text = ds;

await Task.Delay(1000);

operation.Text = "Calculating upload speed.Please wait.";

circularProgress2.IsRunning = true;

circularProgress2.Visible = true;

await Task.Delay(1000);

String us = await CalculateSpeed('u');

//await Task.Delay(1000);

circularProgress2.Visible = false;

uspeed.Text = us;

operation.Visible = false;

labelX1.BringToFront();

labelX1.Visible = true;

await Task.Delay(3000);

Hide();

}

public Task<String> CalculateSpeed(char c)

{

if (c == 'd') **//’d’ stands for download**

return Task.Run(() =>

{

return st.DownloadSpeed();

});

else

return Task.Run(() =>

{

return st.UploadSpeed();

});

}

***Form3:***

1. ***To show the progress bar:***

public async void Progress(Object sender, System.EventArgs e)

{

**//if labelX1.Text contains “Windows was unable to connect”, then show the form for 4 secs and then hide**

if (labelX1.Text.Contains("Windows"))

{

await Task.Delay(4000);

}

else **//else show the progress bar while connecting to the wifi**

{

progressBar1.Visible = true;

await Task.Delay(8000);

}

Hide();

}

***Form5:***

1. ***To set the mode of network (enterprise or normal) :***

public bool setmode

{

get

{

return this.enterprise;

}

set

{

this.enterprise = value;

}

}

1. ***If user enters a security key, then enable the OK button:***

private void key\_TextChanged(object sender, EventArgs e)

{

if (key.Text != "")

OKbutton.Enabled = true;

else

OKbutton.Enabled = false;

}

1. ***If check box is enabled ,then hide the security key entered:***

private void checkBox1\_CheckedChanged\_1(object sender, EventArgs e)

{

if (checkBox1.Checked == true)

key.UseSystemPasswordChar = true;

else

key.UseSystemPasswordChar = false;

}

1. ***If OK button is clicked, then connect to the wifi network using the key provided:***

private void OKbutton\_Click\_1(object sender, EventArgs e)

{

PoziomLabs.Statistics st = new PoziomLabs.Statistics();

PoziomLabs.CurrentWifi cw = new PoziomLabs.CurrentWifi();

if (enterprise) **//in enterprise mode, internet access is needed to verify the user**

{

if (!st.CheckInternetConnection().Equals("Internet Access") || cw.display\_current().Equals("Disconnected"))

{

MessageBox.Show("Please check your internet connection"+Environment.NewLine+"Internet access is needed for verification purpose", "INTERNET ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

Okbuttonclicked = true;

secretkey = key.Text;

Hide();

}

}

else

{

Okbuttonclicked = true;

secretkey = key.Text;

Hide();

}

//f.getkey = true;

}

1. ***If cancel button is clicked:***

private void CANCELbutton\_Click(object sender, EventArgs e)

{

//Form1 f = new Form1();

Okbuttonclicked = false;

Hide();

}

}

}